Distributed Dynamic Fusion Theory and Applications

by

Sam Safavi

A dissertation submitted

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Department of Electrical and Computer Engineering

Tufts University

Advisor: Professor Usman Ahmed Khan

February 2018

Dedicated to my parents for their endless love and support.

Abstract

In this thesis, we study the asymptotic behavior of Linear Time-Varying (LTV) systems that describe fusion in dynamic multi-agent networks. We consider networks that consist of at least one anchor (node with known state) and an arbitrary number of mobile agents that perform a distributed algorithm to achieve a common goal. Due to mobility, the state-update at each agent depends on the availability of neighbors. Agent mobility, thus, leads to an LTV abstraction where the system matrices are random, and can be either stochastic, if the neighborhood at the updating agent does not contain any anchor, or, sub-stochastic, if the neighborhood contains anchors. We refer to the general class of such time-varying fusion algorithms over mobile agents as *Distributed Dynamic Fusion* (DDF). In this context, we study the conditions required on the DDF system matrices such that the dynamic fusion forgets the initial conditions and converges to (a linear combination of) the anchor state(s). To this aim, we partition the sequence of system matrices into non-overlapping *slices*, and by introducing the notion of *unbounded connectivity*, we obtain stability conditions in terms of the slice lengths and some network parameters. In particular, we show that the system is asymptotically stable if the unbounded lengths of an infinite subset of slices grow slower than an explicit exponential rate.

We apply the DDF theory in the design and analysis of the following distributed algorithms: (i) dynamic leader-follower; and (ii) localization in dynamic multi-agent networks. We first consider dynamic leader-follower problem, where the goal for the entire network is to converge to the state of a leader. We assume that at each iteration, one agent exchanges information with nearby nodes and updates its state as a linear-convex combination of the neighboring states. We develop the conditions under which the underlying (sub-) stochastic LTV system converges to the leader state regardless of the agents' initial conditions. We then consider localization in mobile multi-agent networks. We provide a distributed algorithm to localize an arbitrary number of agents moving in a bounded region of interest. In the case of such mobile networks, the main challenge is that the agents may not be able to find nearby agents to implement a distributed algorithm. We address this issue by providing an *opportunistic* algorithm that only implements a linear-convex location update when it lies inside the convex hull of nearby agents and does not update otherwise. We abstract this algorithm as an LTV system, whose system matrices may be stochastic, or sub-stochastic, intermittently, and provide sufficient conditions for the network to track the agents' true locations. By introducing the notion of *virtual convex hull*, we provide an alternative localization algorithm that does not require an agent to lie inside the convex of the neighbors at any given time. We investigate the effects of noise on both

localization algorithms and provide modifications to counter the undesirable effects of noise. Finally, we relate the dimension of motion in the network to the number of anchors required, and show that a network of mobile agents with full degrees of freedom in the motion can be localized precisely, as long as there is *at least one anchor* in the network. To the best of our knowledge, this is a unique property that distinguishes the proposed localization approach in this thesis from the existing algorithms in the literature.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to my PhD advisor, Professor Usman Khan for his encouragement, guidance, and invaluable support during the course of this work. I am grateful for everything you taught me, and I feel incredibly lucky to have had you as my advisor and friend.

I would like to extend my gratitude to my thesis committee members, Professor Jason Rife from Mechanical Engineering Department, Tufts University, Professor Brian Tracey from Electrical and Computer Engineering Department, Tufts University, and Professor José Bento from Computer Science Department, Boston College for taking time to review my thesis. I appreciate the insightful comments and useful suggestions that helped me improve the quality of this work.

Finally, I want to thank my parents for their unwavering love, support, and inspiration not only during my time in graduate school, but throughout every stage of my life. Words cannot describe what you have done for me, and I can't thank you enough.

Chapter 1

Introduction

Stability of linear time-varying (LTV) systems has been a topic of significant interest in a wide range of disciplines including but not limited to modeling and control of dynamical systems, [1–7]. Discrete-time, LTV dynamics can be represented by the following model:

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k, \qquad k \geq 0, \tag{1.1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{P}_k$'s and $\mathbf{B}_k$'s are the system and input matrices, $\mathbf{u}_k \in \mathbb{R}^s$ is the input vector, and k is the discrete-time index. In contrast to the case when the system matrices are time-invariant, i.e., $\mathbf{P}_k = \mathbf{P}, \forall k$, as in many studies related to the above examples, in this thesis we are motivated by the scenarios where the system matrices are time-varying. In particular, we are interested in LTV systems that capture the dynamics of a multi-agent

network, where a collection of autonomous *mobile* agents act towards an objective while interacting in a shared environment via local communications and without any local or central coordinator. In such scenarios, the dynamic system matrices do not only model time-varying neighboring interactions, but in addition, capture the mobility of the agents in the underlying network. We refer to the general class of such time-varying fusion algorithms over mobile agents as *Distributed Dynamic Fusion* (DDF). We assume that the network

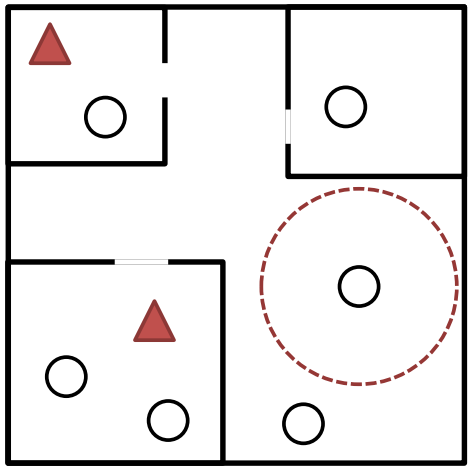

Figure 1.1: DDF environment with 6 agents and 2 anchors.

consists of an arbitrary number of agents and at least one *anchor* (node with known state). A simple DDF environment is illustrated in Fig. 1.1, in which the agents and anchors are represented by black circles and red triangles, respectively, and red dashed circle represents the communication radius of an agent. We assume that an agent can only communicate with its neighbors

(nodes within its communication radius).

In order to design, implement and analyze DDF algorithms we have to deal with some challenges. First of all, since the motion of the agents is arbitrary, the network configuration at any given time is not predictable. Also, it is possible that due to restricted mobility, many agents never directly communicate with the anchors. For examples anchors, as the only reliable source of information in the network, and/or agents might be constrained to different rooms inside a building, see Fig. 1.1. For similar reasons, at many time instances agents may find no neighbor within their communication radii. Finally, the dynamic neighborhood at each agent results into a (linear) time-varying system (if the updates are linear), whose stability (convergence) analysis is not straightforward. In this context, we study the conditions required on the DDF system matrices such that the *dynamic* fusion forgets the initial conditions and converges to (a linear combination of) the anchor state(s). Convergence analysis of such dynamics is equivalent to studying the stability of the underlying LTV system, Eq. (1.1).

For linear time-invariant (LTI) systems, a necessary and sufficient condition for stability is that the *spectral radius*, i.e., the largest eigenvalue in magnitude, of the system matrix is strictly less than one. In contrast, the DDF algorithms over mobile agents result into a time-varying system, Eq. (1.1), where a system matrix, $\mathbf{P}_k$, at any time k is non-negative, and can be: (i) ***identity***, if no agent

is able to update its state; (ii) ***stochastic***[1], if the updating agent updates its state as a weighted average of the states of the neighboring agents not including an anchor, i.e., divides the total weight of 1 among the agents in its neighborhood; or, (iii) ***sub-stochastic***[1], if the total weight of 1 is divided among both agents and anchors. In such settings, it is straightforward to verify that the resulting LTV system may be such that the spectral radius, $\rho(\mathbf{P}_k)$, of the system matrices follow $\rho(\mathbf{P}_k) = 1,\ \forall k$.

A theoretical approach to study the asymptotic stability of the LTV dynamics in Eq. (1.1) is to find *the Joint Spectral Radius* (JSR) of the associated family of system matrices. Joint spectral radius is the generalization of the classical notion of spectral radius in LTI systems. Although the JSR characterizes the stability of LTV systems, the problem of determining whether the JSR of a set of matrices is less than one is undecidable, [8].

Motivated by DDF, in the *first part* of this thesis we provide an alternative approach to study the asymptotic behavior of such dynamics, which does not require the computation or approximation of the joint spectral radius. To this aim, we partition the entire chain of system matrices into non-overlapping *slices*. We define a slice, $\mathbf{M}_t$, as a product of consecutive system matrices, $\mathbf{P}_k$'s, such that: (i) each slice has a subunit infinity norm; and, (ii) the entire set of system matrices is covered by non-overlapping slices. As we will show

[1]Refer to Section 1.2 for the exact definition.

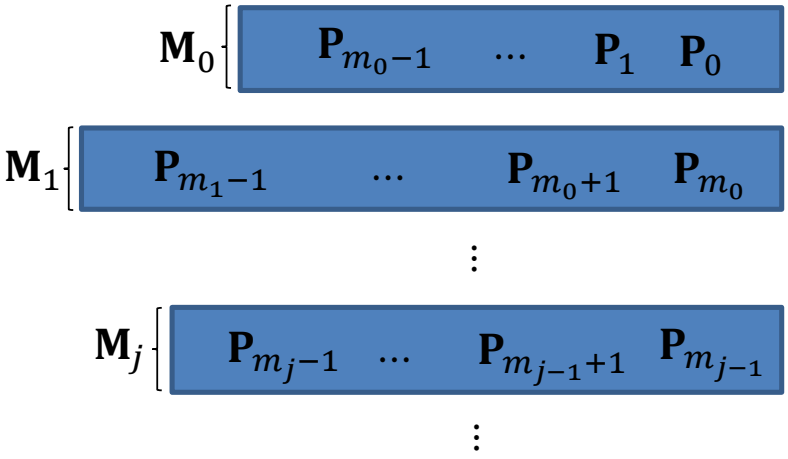

Figure 1.2: Slices cover the entire set of system matrices.

in Chapter 2, a slice must be initiated by a sub-stochastic system matrix and is terminated after all row sums become strictly less than one. An important aspect of our analysis lies in the study of the slice lengths. We show that longer slices may have an infinity norm that is closer to one as compared to shorter slices. Clearly, if one can show that each slice norm is subunit (with a uniform upper bound of < 1) then one further has to guarantee an infinite number of such slices to ensure stability. The aforementioned argument naturally requires slices of finite length, as finite slices covering infinite (system) matrices lead to an infinite number of slices. Note that guaranteeing a sharp upper bound on the length of every slice may not be possible for certain network configurations. To address such configurations, we characterize the rate at which the slices (not necessarily in an order) grow large such that the LTV stability is not

disturbed and fusion is still achievable. We introduce the notion of *unbounded connectivity*, and show that the time-intervals over which the the slices are completed do not have to be bounded as long as they grow slower than an explicit exponential rate.

In the *second part* of this thesis, we apply the theoretical results from the first part in the design and analysis of the following DDF applications:

(i) dynamic leader-follower; and,

(ii) localization in dynamic networks.

We first consider *dynamic leader-follower* problem where the goal for the entire network is to converge to the state of a leader or a linear combination of leader states in case of multiple leaders. We assume that the agents move randomly in a bounded region of interest and have a limited communication range. We assume that only one agent at each iteration exchanges information with nearby nodes and updates its state as a linear-convex combination of the neighboring states.

It is well-known that under mild conditions on network connectivity the agent (follower) states converge to the state of the leader (assuming only one leader in the network). However, the neighboring interactions change over time if the nodes are mobile. In the case of possibly random motion over the network, an agent may not find neighbors at all times and no agent may

communicate with a leader at any given time. This leads to different state-update scenarios, which can be abstracted as an LTV system. We develop the conditions under which this LTV system converges to the leader state regardless of the agents' initial conditions. In order to develop the results for (sub-) stochastic matrices, we use the notion of non-overlapping slices; one complete slice implies information propagation from the leader to every other agent in the network either directly, when an agent communicates with the anchor, or indirectly, when an agent communicates with another agent that has previously communicated with the anchor. Clearly, if this information dissemination is completed in a bounded time, i.e., the slices have a uniform bound on their lengths, then the information from the leader reaches the entire network infinitely often and the asymptotic convergence follows. However, we show that the dynamic leader-follower algorithm converges even when the slice lengths are unbounded; what is required is that the slice lengths do not grow larger at a rate faster than a certain exponential growth. In other words, the information from the leader reaches the rest of the network in an unbounded number of steps, the rate of which is explicitly characterized.

As another application of DDF, we then consider *localization problem in dynamic networks*. Localization is a well-studied problem, which refers to a collection of algorithms that estimate the location of static or mobile nodes in a network. Relevant applications include traffic control, industrial automation,

robotics, environment monitoring, 5th generation mobile networks, and the Internet of Things (IoT) among others, [9–23]. The majority of localization algorithms in mobile networks are Bayesian methods[2] that use the recursive Bayes' rule to estimate the likelihood of an agent's location. Due to the non-linear nature of the involved conditional probabilities, the Bayesian solutions are generally intractable and cannot be determined analytically. Solutions involving extended Kalman filters, [24–33], particle filters, and sequential Monte Carlo methods, [34–45], have been proposed to address the associated non-linearities and the intractable computation of the probability distributions. However, these approaches end up being sub-optimal and highly susceptible to the agents' initial guesses of their locations, while requiring a relatively high density of anchors to achieve accurate location estimates.

As an alternative to the traditional non-linear approaches, in this thesis, we provide a distributed *linear* algorithm to solve localization in a network of mobile agents in the absence of any prior estimates of the initial locations. This linear framework is not to be interpreted as a linearization of an existing nonlinear algorithm. Instead, the nonlinearity from range to location is embedded in an alternate representation provided by the *barycentric coordinates*. We assume that the network consists of at least one (possibly mobile) anchor

[2]We provide an overview of the predominant approaches for localization in mobile networks in Appendix A.

whose location is known at all times, and each mobile agent in the network can measure a noisy version of its motion and the distances to the neighboring nodes (agents and/or anchors), when available. We develop a *linear-convex* and distributed set of iterations, and show that they converge to the true locations regardless of the agents' initial conditions. Our proposed localization algorithm in $\mathbb{R}^m$ can be described as follows. At the beginning, all agents are

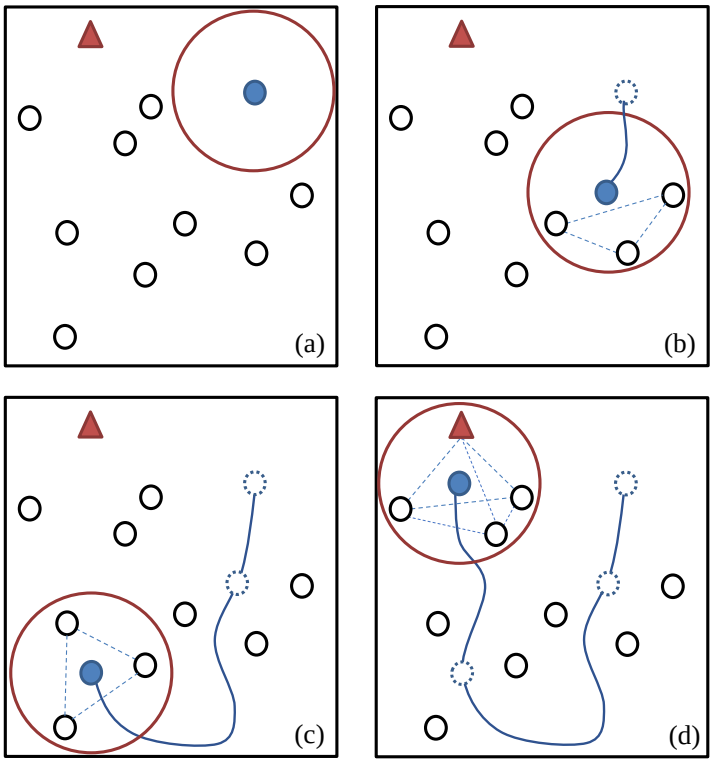

Figure 1.3: Update scenarios for localization in $\mathbb{R}^2$.

assigned with arbitrary random estimates of their initial locations. To perform an update in $\mathbb{R}^m$, agent i needs to lie inside the convex hull of $m+1$

neighboring nodes[3]. Fig. 1.3 shows different update scenarios of the proposed localization algorithm in $\mathbb{R}^2$; in Fig. 1.3 (a), the blue agent does not find any neighbors within its communication radius and does not perform an update. In Fig. 1.3 (b), the blue agent finds $m + 1 = 3$ neighbors that does not pass the inclusion test. Hence, no update occurs at this time. On the other hand, Fig. 1.3 (c) and (d) show two possible update scenarios, where the blue agent updates its location estimate as a weighted sum of the states of its neighboring agents, or a combination of the states of agents and anchors, respectively.

The convex set of neighbors keeps changing over time as the agents are mobile. The dynamic neighborhood at each agent results into a linear time-varying system, which comprises of system matrices that may be (i) ***identity***: if no agent is able to perform an update; (ii) ***stochastic***: if the updating agent lies inside the convex hull of $m + 1$ other agents; or, (iii) ***sub-stochastic***: if the updating agent lies inside the convex hull of a combination of agents and anchors. Establishing the asymptotic behavior of the resulting LTV system is non-trivial. We apply results from the first part of the thesis to study LTV convergence by partitioning the entire set of system matrices into slices and relating the convergence rate to the slice lengths. In particular, we show that agent location estimates are refined as the procedure continues and the

[3]To this end, we utilize a simple *inclusion test*, described in the Appendix B.1, which only requires the mutual distance information and allows an agent to check if it lies inside the convex hull of a set of neighbors.

algorithm converges if the slice lengths do not grow faster than a certain exponential rate. In the context of the proposed localization algorithm, one slice can be cast as a time interval, over which the reliable location information is propagated from the anchor(s) across the entire network. In other words, an increase in the slice length captures a slow information propagation in the network. In this regard, initiation of a slice corresponds to an update, in which an agent receives the information directly from an anchor, and a slice is terminated after all agents receive the information from the anchors either directly or indirectly. In this context, the aforementioned exponential rate corresponds to a rate, at which the information propagation may deteriorate in the network such that all agents asymptotically track their true locations.

We then extend the aforementioned localization algorithm by allowing the agents to perform location updates without physically lying inside the convex hull of their neighbors. We achieve the notion of a *virtual convex hull*, which allows an agent to perform a location update without passing the inclusion test at any given time. To this aim, we provide a *geometric approach*, which allows each agent to: (i) continually update the distances to the locations where it has exchanged information with the other nodes in the past; and (ii) measure the distance between a neighbor and any such locations. Based on this approach, we provide a linear algorithm to find the locations of an arbitrary number of mobile agents when they follow some convexity in their deployment

and motion in $\mathbb{R}^2$. We abstract the corresponding localization algorithm as an LTV system with (sub-) stochastic system matrices, and apply the results from the first part of the thesis to provide sufficient conditions on the convergence of the algorithm to the true agent locations.

Since the proposed localization schemes are based on the motion and the distance measurements, we evaluate the performance of both algorithms when these parameters are corrupted by noise, and provide modifications to counter the undesirable effects of noise. Moreover, we investigate what role does motion play in the behavior of the localization algorithms, and provide necessary conditions in terms of the number of nodes and the dimensions of motion required by our approach.

The results in this thesis have been published in the following peer-reviewed journals and conferences, [46–54]. In the following, we enlist the main contributions of this thesis.

1.1 Contributions

We now summarize the main contributions of this thesis.

1. **Stability of (sub-) stochastic LTV systems (Chapter 2):** We study asymptotic stability of linear time-varying systems with randomly chosen, stochastic or sub-stochastic system matrices and provide an al-

ternative analysis approach in contrast to the computation or approximation of the joint spectral radius. Motivated by applications in distributed dynamic fusion, we impose some mild regularity conditions on the elements of time-varying system matrices, and provide sufficient conditions under which the asymptotic stability of the underlying LTV system is guaranteed. To this aim, we partition the sequence of system matrices into non-overlapping slices, and by introducing the notion of unbounded connectivity, we obtain stability conditions in terms of the slice lengths and some network parameters. We use the infinity norm to characterize the asymptotic stability and provide upper bounds on the infinity norm of each slice as a function of its length. We show that asymptotic stability is guaranteed not only in the trivial cases where all (or an infinite subset) of the slices have a bounded length, but also if there exists an infinite subset of slices whose (unbounded) lengths do not grow faster than a particular exponential growth.

2. **Distributed dynamic leader-follower (Chapter 3):** We study the leader-follower problem in mobile multi-agent networks, when the nature of communications within the network is random, i.e., no agent (follower) may be able to communicate with the neighbors at all times and some agents may not be able to communicate with the leader at any give time.

We capture the mobility in the leader-follower algorithm by abstracting it as an LTV system with randomly appearing stochastic or sub-stochastic system matrices. In particular, a mobile agent, moving randomly in a bounded region, updates its state when it finds neighbors; and does not update when it is not in the communication range of any other node. In this context, we develop certain regularity conditions on the system and input matrices such that each follower converges to the leader state. In contrast to the existing results, we show that a bounded length on the slices, capturing the dissemination of information from the leader to the followers, is not required; as long as the slice lengths are finite and do not grow faster than a certain exponential rate. Although we focus on the networks with only one leader, the results can be easily generalized to any network with multiple leaders, where the followers converge to a linear-convex combination of the leader states.

3. **Distributed localization in dynamic networks (Chapter 4):** We develop a distributed algorithm to localize a network of agents that move arbitrarily in a bounded region of interest. In the case of such mobile networks, the main challenge is that the agents may not be able to find nearby nodes to implement a distributed algorithm. We address this issue by providing an *opportunistic* algorithm that only implements a

location update when there are nearby agents and does not update otherwise. To localize a network of mobile agents in $\mathbb{R}^m$, we provide a simple linear-convex update, which is based on barycentric coordinates. In this context, the main contribution of this work is to develop a *linear* framework for localization that enables us to circumvent the challenges posed by the predominant nonlinear approaches to localization problem. This linear framework is not to be interpreted as a linearization of an existing nonlinear algorithm. Instead, the nonlinearity from range to location is embedded in an alternate representation provided by the barycentric coordinates. We abstract the corresponding localization algorithm as an LTV system and show that it asymptotically converges to the true locations of the agents. We first focus on the noiseless case, where the distance and motion vectors are known (measured) perfectly, and provide sufficient conditions on the convergence of the algorithm. We then evaluate the performance of the algorithm in the presence of noise and provide modifications to counter the undesirable effects of noise.

4. **Distributed localization via virtual convex hull (Chapter 5):** We provide an alternative distributed algorithm in $\mathbb{R}^2$, when no agent may ever lie inside an actual physical convex hull. We first provide a geometric framework, which allows an agent to keep track of the distances to

any previously visited nodes, and find the distance between a neighbor and any virtual location where it has exchanged information with other nodes in the past. Since agents are mobile, they may not be able to find neighbors to perform distributed updates at any time. To avoid this issue, we introduce the notion of a *virtual convex hull*, which forms the basis of our second localization algorithm in mobile networks. We abstract the algorithm as an LTV system with (sub-) stochastic matrices, and show that it converges to the true locations of agents under some mild regularity conditions on update weights. We evaluate the performance of the algorithm in presence of noise and provide modifications to the proposed algorithm to address noise on motion and on distance measurements.

5. **Localization in mobile networks using one anchor (Chapters 4 and 5):** In contrast to the existing localization algorithms in the literature, we show that our algorithms precisely track a mobile network as long as the network contains *at least one* anchor. We show that the remaining degrees of freedom are, in fact, delivered by the motion.

In the next section, we provide some definitions and standard notations that will be used in the rest of the thesis.

1.2 Preliminaries and notations

We use the following notations in the rest of this thesis. We use lowercase bold letters to denote vectors and uppercase bold letters to denote matrices. We denote by $\mathbf{x}_i^k$, the i-th component of a vector $\mathbf{x}$ at time k. For a time-varying matrix, $\mathbf{A}_k$, we denote the (i,j)-th element of matrix at time k by $\mathbf{A}_k^{ij} = (\mathbf{A}_k)_{i,j}$. The matrix, $\mathbf{I}_n$, represents the $n \times n$ identity, and $\mathbf{1}_n$ and $\mathbf{0}_n$ are the n-dimensional vector of all 1's and 0's. We denote the cardinality, i.e., the number of elements in the set S, by $|S|$. We now make the following definitions to clarify what we mean by (sub-) stochasticity throughout this thesis:

Definition 1. *A non-negative, stochastic matrix is such that all of its row sums are one.*

Definition 2. *A non-negative, sub-stochastic matrix is such that it has at least one row that sums to strictly less than one and every other row sums to at most one.*

Note that in the related literature the above matrices may be referred to as row stochastic and row sub-stochastic matrices, respectively. We finish this section with the following definitions that will be used in the following chapters.

Definition 3. *For an $m \times n$ matrix, $\boldsymbol{A}$, the infinity norm, $\|\boldsymbol{A}\|_\infty$, is defined*

as the maximum of the absolute row sum of a matrix, i.e.,

$$\|\boldsymbol{A}\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |(\boldsymbol{A})_{i,j}|.$$

Definition 4. *For a square matrix, $\boldsymbol{A}$, the spectral radius, $\rho(\boldsymbol{A})$, is defined as the absolute value of the largest eigenvalue.*

Definition 5. *The LTV system represented in Eq.* (1.1) *is asymptotically stable (or convergent) if for any initial condition, $\boldsymbol{x}_0$, $\lim_{k\to\infty} \boldsymbol{x}_k$ is bounded and convergent.*

Definition 6. *The LTV system represented in Eq.* (1.1) *is absolutely asymptotically stable (or zero-convergent) if for any initial condition, $\boldsymbol{x}_0$, we have*

$$\lim_{k\to\infty} \boldsymbol{x}_k = 0.$$

1.3 Summary

In this chapter, we motivate distributed dynamic fusion as a general class of time-varying fusion algorithms with random (sub-) stochastic system matrices, and summarize the contributions made in this thesis.

Chapter 2

Asymptotic stability of (sub-) stochastic LTV systems

In this chapter, we investigate the behavior of linear time-varying (LTV) systems with randomly appearing, (sub-) stochastic system matrices. Motivated by dynamic fusion over mobile networks, we develop conditions on the system matrices that lead to asymptotic stability of the underlying LTV system. By partitioning the sequence of system matrices into *slices*, we obtain the stability conditions in terms of slice lengths and introduce the notion of *unbounded connectivity*, i.e., we show that the time-intervals, over which the multi-agent network is connected, do not have to be bounded as long as they do not grow faster than a certain exponential rate. In the next chapters, we apply the above analysis to derive the asymptotic behavior of a dynamic leader-follower

algorithm and to study the convergence of two distributed algorithms for localization in mobile networks.

2.1 Introduction

Stability of linear time-varying systems has been a topic of significant interest in a wide range of disciplines including but not limited to mathematical modeling and control of dynamical systems, [1–7]. Discrete-time, LTV dynamics can be represented by the following model:

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k, \qquad k \geq 0, \tag{2.1}$$

in which $\mathbf{x}_k$ is the state vector, $\mathbf{u}_k$ is the input vector, and $\mathbf{P}_k$ and $\mathbf{B}_k$ represent the system and input matrices, respectively.

We are interested in studying the asymptotic behavior of LTV systems that describe fusion in, e.g., *mobile* multi-agent networks in the presence of *anchors* that are nodes with known states. Due to mobility, the state-update at each agent depends on the availability of neighbors. Agent mobility, thus, leads to an LTV abstraction where the system matrices are random, and can be either *stochastic*, if the neighborhood at the updating agent does not contain any anchor, or, *sub-stochastic*, if the neighborhood contains anchors. We refer to the general class of such algorithms over mobile agents as *Distributed Dy-*

namic Fusion (DDF). Relevant examples include leader-follower, [55], sensor localization, [56], and consensus-based control, [57].

In this context, we study the conditions required on the DDF system matrices such that the agents: (i) forget the initial conditions; and, (ii) converge to some linear combination of the anchor states. If we ignore the latter for the sake of discussion, the DDF problem is to ensure that the underlying LTV system asymptotically converges to zero in the absence of anchors. Hence, we first describe the conditions under which this stability is guaranteed. Adding anchors, one has to additionally show that the convergence to (a function of) the anchor states is possible.

For a linear time-invariant (LTI) system,

$$\mathbf{x}_{k+1} = \mathbf{P}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \qquad k \geq 0, \tag{2.2}$$

a necessary and sufficient condition for stability is that the *spectral radius* of the system matrix is strictly less than one. A well-known result from matrix theory is that if the (time-invariant) system matrix, $\mathbf{P}$, is irreducible[1] *and* sub-stochastic, sometimes referred to as *uniformly sub-stochastic*, [58], the spectral radius of $\mathbf{P}$ is subunit, and in Eq. (2.2),

$$\lim_{k\to\infty} \mathbf{x}_k \to 0,$$

in the absence of external inputs.

[1] A square matrix is irreducible if and only if its associated digraph is strongly connected.

In contrast, the DDF algorithms over mobile agents result into a time-varying system, Eq. (2.1), where a system matrix, $\mathbf{P}_k$, at any time k is non-negative, and can be:

(i) ***identity***, if no agent is able to update its state;

(ii) ***stochastic*** if the updating agent divides the total weight of 1 among the agents in its neighborhood; or,

(iii) ***sub-stochastic***, if the total weight of 1 is divided among both agents and anchors.

In addition, it is straightforward to verify that in DDF algorithms, the resulting LTV system may be such that the spectral radius, $\rho(\mathbf{P}_k)$, of the system matrices follow

$$\rho(\mathbf{P}_k) = 1, \qquad \forall k.$$

This is the case, for example, when at each iteration some agents update with their neighboring agents and the remaining agents stick to their past states.

Asymptotic stability for LTV systems may be characterized by the *joint spectral radius* of the associated family of system matrices. Given a finite set of matrices, $\mathcal{M} = \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$, the joint spectral radius of the set $\mathcal{M}$, was introduced by Rota and Strang, [59], as a generalization of the classical notion

of spectral radius, with the following definition:

$$\rho(\mathcal{M}) := \lim_{k\to\infty} \max_{\mathbf{A}\in\mathcal{M}_k} \|\mathbf{A}\|^{\frac{1}{k}},$$

in which $\mathcal{M}_k$ is the set of all possible products of the length $k \geq 1$, i.e.

$$\mathcal{M}_k = \{\mathbf{A}_{i_1}\mathbf{A}_{i_2}\dots\mathbf{A}_{i_k}\}.$$

Joint spectral radius (JSR) represents the maximum growth rate that can be achieved by forming arbitrary long products of the matrices taken from the set $\mathcal{M}$. It turns out that the asymptotic stability of the LTV systems, with system matrices taken from the set $\mathcal{M}$, is guaranteed, [60], if and only if

$$\rho(\mathcal{M}) < 1.$$

Although the JSR characterizes the stability of LTV systems, the problem of determining $\rho(\mathcal{M}) \leq 1$ is undecidable, [8]. In [61], Tsitsiklis and Blondel prove that, it is NP-hard to decide if all possible products of *two* given matrices are stable. Naturally, much of the existing literature has focused on JSR approximations, [8, 60–63].

To develop the asymptotic stability arguments, our approach does not require either the computation or an approximation of the joint spectral radius. Instead, we partition the infinite set of (non-deterministic) system matrices into non-overlapping slices–a slice is defined as the smallest product of (con-

secutive) system matrices such that: (i) every row sum in a slice is strictly less than one; and, (ii) the slices cover the entire sequence of system matrices.

Decentralized algorithms such as agreement and consensus are relevant here, see, e.g., [64–70]; Ref. [64], for example, derives convergence to the anchor state, if there exists an infinite sequence of contiguous, non-empty, and *bounded time-intervals* such that across each interval, each agent is connected to the anchor. Ref. [65] extends [64] to directed graphs and shows that asymptotic consensus is achieved if there exist an infinite set of *uniformly bounded intervals* within which the union of the graphs has a spanning tree. Ref. [66] characterizes the limiting matrix for the product of stochastic matrices in terms of the topology of the infinite flow graph. Ref. [67] considers a distributed feedback controller and shows that consensus can be achieved if there exists an infinite sequence of contiguous, nonempty, *uniformly bounded* time-intervals for which the union of communication topologies has a spanning tree. Ref. [68], on the other hand, provides a surplus-based averaging algorithm and shows that a network of agents achieves uniform average consensus if and only if the dynamic digraph is jointly strongly-connected in a periodic manner. Relevant work also includes [69] that provides the conditions for asymptotic stability and reachability of consensus in a discrete-time, switched system, where the system matrices belong to a *finite set* that share the common eigenvector of all constants. Finally, Ref. [70] considers consensus problem in the context

of distributed optimization over connected graphs, where different agents of the distributed implementation communicate to agree on the final solution, and studies the rate at which an agreement can be achieved. Stability of LTV systems is also studied for infinite products of matrices, see, e.g., [71–74].

The main contributions of the approach presented in this chapter are as follows. First, the system matrices in our formulation do not belong to a finite or a countable set. Second, we derive *explicit bounds* on the infinite-norm of the aforementioned slices. Third, the system matrices can be either stochastic or sub-stochastic, intermittently; thus, their product is no longer a group under multiplication and requires careful weight design. A significant aspect of our analysis lies in the study of the slice lengths. As guaranteeing a sharp upper bound on the length of every slice may not be possible, we introduce the notion of *unbounded connectivity*; we show that the time-intervals, over which each agent is connected to the anchor, do not have to be bounded; as long as they do not grow faster than an *explicit exponential rate*. The notion of unbounded connectivity is in stark contrast to the literature described above where sharp (time-interval) bounds are assumed. In other words, a longer slice may capture a slow information propagation in the network; characterizing the aforementioned growth is equivalent to deriving the rate at which the information propagation may deteriorate in a network such that fusion is still achievable.

We now describe the rest of this chapter. We formulate the problem in Section 2.2. We examine the infinite product of (sub-) stochastic matrices in Section 2.3, and study the stability of (sub-) stochastic LTV systems in Section 2.4. Simulation results are provided in Section 2.5 and finally, Section 2.6 concludes this chapter.

2.2 Problem formulation

In this chapter, we study the asymptotic stability of the following linear time-varying dynamics:

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k, \qquad k \geq 0, \tag{2.3}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{P}_k \in \mathbb{R}^{n \times n}$ is the system matrix, $\mathbf{B}_k \in \mathbb{R}^{n \times s}$ is the input matrix, $\mathbf{u}_k \in \mathbb{R}^s$ is the input vector, and k is the discrete-time index. We consider $\mathbf{P}_k$ at each k to be either identity, stochastic or sub-stochastic. We are interested in deriving the conditions on the corresponding system matrices under which the LTV dynamics in Eq. (2.3) forget the initial condition, $\mathbf{x}_0$, and converge to some function of the input vector, $\mathbf{u}_k$. The motivation behind this investigation can be cast in the following context.

Distributed Dynamic Fusion: Consider a network of $n+s$ mobile nodes moving arbitrarily in a bounded region. The network consists of n agents in

the set Ω, whose states[2], $\mathbf{x}_k^i \in \mathbb{R}, i \in \Omega$, are updated as a function of the neighbors' states to obtain some relevant function of s anchors in the set κ, whose states, $\mathbf{u}_k^j = \mathbf{u}^j \in \mathbb{R}$, $j \in \kappa$, are constant and do not change over time, see, e.g., [55, 56, 75, 76]. Since the motion of each node is arbitrary, the network configuration at any time k is unpredictable. Let $\mathcal{N}_k^i$ denote the set of neighbors[3] of agent i at time k, with $\mathcal{D}_k^i \triangleq \{i\} \cup \mathcal{N}_k^i$. We assume that at each time k, only one agent, say i, updates its state, $\mathbf{x}_k^i$. Since the underlying graph is dynamic, the updating agent i implements one of the following updates:

(i) ***no update***, if $\mathcal{N}_k^i = \emptyset$, i.e., agent i has no neighbors at time k:

$$\mathbf{x}_{k+1}^i = \mathbf{x}_k^i. \tag{2.4}$$

(ii) ***stochastic update***, if $\mathcal{N}_k^i \cap \kappa = \emptyset$, i.e., agent i has no neighboring anchor at time k:

$$\mathbf{x}_{k+1}^i = \sum_{l \in \mathcal{D}_k^l} (\mathbf{P}_k)_{i,l} \mathbf{x}_k^l, \tag{2.5}$$

in which $(\mathbf{P}_k)_{i,l}$, the (i,l)-th element of system matrix at time k, is the weight agent i assigns to the state of the l-th agent.

[2]In general, the i-th state at time k can be either a scalar or a row vector. In the remainder of this chapter, we consider scalar states.

[3]Note that a neighbor is defined as any node that lies within the communication radius, r, of an agent.

(iii) ***sub-stochastic update***, if $\mathcal{N}_k^i \cap \kappa \neq \emptyset$, i.e., agent i has at least one anchor[4] as a neighbor:

$$\mathbf{x}_{k+1}^i = \sum_{l \in \mathcal{D}_k^l \cap \Omega} (\mathbf{P}_k)_{i,l} \mathbf{x}_k^l + \sum_{l \in \mathcal{D}_k^l \cap \kappa} (\mathbf{B}_k)_{i,j} \mathbf{u}_k^j, \tag{2.6}$$

in which $(\mathbf{B}_k)_{i,j}$, the (i,j)-th element of input matrix at time k, is the weight agent i assigns to the state of the j-th anchor.

For every other agent, $l \neq i$, at time k we have

$$\mathbf{x}_{k+1}^l = \mathbf{x}_k^l. \tag{2.7}$$

Note that the way the non-zero elements of $\mathbf{P}_k$ and $\mathbf{B}_k$ matrices, i.e., the weights assigned to the states of the neighboring nodes at the time of communication, are chosen depends on the DDF algorithm and the corresponding update equation. For example, in the context of localization such weights come from the Barycentric coordinates, see Chapter 4 for more details.

Assumptions

We now enlist the assumptions:

A0: When the updating agent, i, has no neighboring anchor, we have

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} = 1, \tag{2.8}$$

resulting in a stochastic $\mathbf{P}_k$.

[4]We assume that the agents can distinguish anchors from other agents at the time of communication.

A1: When the updating agent, i, has no anchor but at least one agent as a neighbor, we assume

$$0 < \beta_1 \leq (\mathbf{P}_k)_{i,l} < 1, \qquad \forall l \in \mathcal{D}_k^i,\ \beta_1 \in \mathbb{R}. \tag{2.9}$$

A2: When the updating agent updates with an anchor, we assume

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} \leq \beta_2 < 1,\ \beta_2 \in \mathbb{R}, \tag{2.10}$$

resulting in a sub-stochastic system matrix, $\mathbf{P}_k$.

Note that β_1 and β_2 are time-independent. Also note that the update over the anchors, $\mathcal{N}_k^i \cap \kappa$, in Eq. (2.6), follows

$$0 < \alpha \leq (\mathbf{B}_k)_{i,j}, \qquad \forall j \in \mathcal{N}_k^i \cap \kappa. \tag{2.11}$$

Also, if we have

$$\sum_l (\mathbf{P}_k)_{i,l} + \sum_j (\mathbf{B}_k)_{i,j} = 1, \tag{2.12}$$

as in leader-follower, [55], or sensor localization, [56], Eq. (2.11) leads to Eq. (2.10).

Which of the four updates in Eqs. (2.4)–(2.6) is applicable depends on being able to satisfy the corresponding assumptions (**A0**–**A2**), *in addition* to the neighborhood configuration. Indeed, letting $\mathbf{P}_k = (\mathbf{P}_k)_{i,l}, \mathbf{B}_k = (\mathbf{B}_k)_{i,j}$, and

$$\mathbf{x}_k = \left[x_k^1,\ \ldots\ , x_k^n\right]^\top,\quad \mathbf{u}_k = \left[u_k^1,\ \ldots\ , u_k^s\right]^\top,$$

results into Eq. (2.3), where the matrices, $\mathbf{P}_k$, are either identity, stochastic or sub-stochastic, depending on the update. We now briefly explain the the intuition behind the above assumptions. Eq. (2.10) provides an *upper bound on unreliability* when there is a neighboring anchor, thus restricting the unreliable information added by an agent. Eq. (2.11) provides a *lower bound on reliability* ensuring that an anchor always contributes a certain amount of information. When there is no neighboring anchor, Eq. (2.9) guarantees that the agents retain the information they may have received from the anchor by putting a non-zero self-weight.

2.3 Infinite product of (sub-) stochastic matrices

In this section, we study the infinite product,

$$\lim_{k\to\infty} \mathbf{P}_k\mathbf{P}_{k-1}\dots\mathbf{P}_0,$$

where $\mathbf{P}_k$ is the system matrix at time k, as defined in Section 2.2. We are interested in establishing the stability properties of this infinite product. Studying the joint spectral radius is prone to many challenges as described in Section 2.1; we use the infinity norm as defined in Chapter 1. The infinity norm of $\mathbf{P}_k$ is 1 for all k since each system matrix has at most one sub-stochastic

row. To establish a subunit infinity norm, we partition the entire chain of system matrices into non-overlapping *slices* and show that each slice has an infinity norm strictly less than one. Let one of the slices be denoted by $\mathbf{M}$ with length $|\mathbf{M}|$ and, without loss of generality, index the matrices within $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{P}_{|\mathbf{M}|}\mathbf{P}_{|\mathbf{M}|-1}\mathbf{P}_{|\mathbf{M}|-2}\dots\mathbf{P}_3\mathbf{P}_2\mathbf{P}_1. \tag{2.13}$$

Slices are initiated by strictly sub-stochastic system matrices[5], and terminated

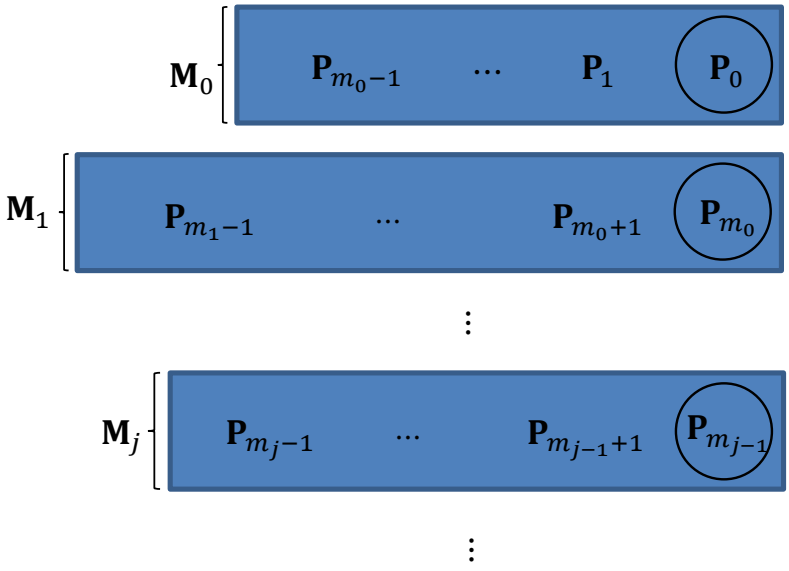

Figure 2.1: Slice representation.

after all row sums are strictly less than one. Slice representation is depicted in Fig. 2.1, where the rightmost system matrices (encircled in Fig. 2.1) of each slice, i.e., $\mathbf{P}_0,\ \mathbf{P}_{m_0},\ \dots,\ \mathbf{P}_{m_{j-1}}\dots$, are sub-stochastic. The length of a slice is

[5]In the context of DDF, sub-stochasticity of the i-th row in $\mathbf{P}_k$ corresponds to an update at agent i that involves an anchor.

defined as the number of matrices forming the slice, and using the indices in Fig. 2.1, for the j-th slice length we have

$$|\mathbf{M}_j| = m_j - m_{j-1}, \qquad m_{-1} = 0.$$

Using slice notation, we introduce a new index, t, and study the infinite product of slices,

$$\lim_{t\to\infty} \mathbf{M}_t \mathbf{M}_{t-1} \dots \mathbf{M}_0,$$

instead of the product of $\mathbf{P}_k$'s.

We define an update in the i-th row of a system matrix, $\mathbf{P}_k$, as a *success* if it decreases the i-th row sum in $\mathbf{P}_k$, which was previously stochastic. Each success, thus, adds a *new* sub-stochastic row to a slice, and n such successful updates are required to complete a slice. In this argument, we assume that a row sum that becomes less than one remains less than one, which is not true, in general, after successive multiplication with sub-stochastic matrices. Thus, we will derive the explicit conditions under which the sub-stochasticity of a row is preserved. Assuming only one update per iteration, we start with the following lemma:

Lemma 1. *For the infinity norm of a slice to be less than one, it has to contain at least one sub-stochastic update.*

Proof. Since any set of stochastic matrices form a *group* under multiplication [77], a slice without a sub-stochastic update will be a stochastic matrix

whose infinity norm is 1. □

Construction of a slice may be motivated as follows. Partition the rows in an arbitrary $\mathbf{P}_k$ into two distinct sets: set $\mathcal{I}$ contains all sub-stochastic rows, and the remaining (stochastic) rows form the other set, $\mathcal{U}$. We initiate each slice with the first success, $|\mathcal{I}| = 1,\ |\mathcal{U}| = n-1$, and terminate it *after* the n-th success, $|\mathcal{I}| = n,\ |\mathcal{U}| = 0$. Between the n-th success in the current slice, say $\mathbf{M}_j$, and the first success in the next slice, $\mathbf{M}_{j+1}$, all we can have are stochastic matrices that must preserve the sub-stochasticity of each row. These matrices belong to slice $\mathbf{M}_j$. The next lemmas show how a stochastic row becomes sub-stochastic in a slice, $\mathbf{M}$, and how sub-stochasticity is preserved. We index $\mathbf{P}_k$'s in a slice, $\mathbf{M}$, by $\mathbf{P}_{|\mathbf{M}|}, \ldots, \mathbf{P}_2, \mathbf{P}_1$ to simplify notation, and define the product of all system matrices up to time k in a slice as

$$\mathbf{J}_k = \mathbf{P}_k \mathbf{P}_{k-1} \ldots \mathbf{P}_2 \mathbf{P}_1, \qquad 0 < k \leq |\mathbf{M}|. \tag{2.14}$$

Lemma 2. *Suppose the i-th row of $\boldsymbol{J}_k$ is stochastic at index k of a given slice, $\boldsymbol{M}$, and $\boldsymbol{P}_{k+1}$ is the next system matrix. Row i in $\boldsymbol{J}_{k+1}$ is sub-stochastic with either: (i) a sub-stochastic update at the i-th row of $\boldsymbol{P}_{k+1}$; or, (ii) a stochastic update at the i-th row of $\boldsymbol{P}_{k+1}$, such that $\exists j \in \mathcal{I}_k$, with $(\boldsymbol{P}_{k+1})_{i,j} \neq 0$, where $\mathcal{I}_k$ is the set of sub-stochastic rows in $\boldsymbol{J}_k$.*

Proof. For the sake of simplicity, let $\mathbf{P} \triangleq \mathbf{P}_{k+1}, \mathbf{J} \triangleq \mathbf{J}_k$, in the following.

Updating the i-th row at index $k+1$ leads to

$$\mathbf{P}_{k+1} \triangleq \mathbf{P} = \begin{bmatrix} \mathbf{I}_{1:i-1} & & & \\ (\mathbf{P})_{i,1} & (\mathbf{P})_{i,2} & \dots & (\mathbf{P})_{i,n} \\ & & & \mathbf{I}_{i+1:n} \end{bmatrix}, \tag{2.15}$$

where $\mathbf{I}_{1:n}$ is the $n \times n$ identity. Thus, $\mathbf{P}_{k+1}$ in Eq. (2.15) is the same as identity matrix, except at the i-th row, whose elements are $(\mathbf{P})_{i,1}, (\mathbf{P})_{i,2}, \dots, (\mathbf{P})_{i,n}$. The i-th row after this update is

$$(\mathbf{PJ})_{i,j} = \sum_{m=1}^{n} (\mathbf{P})_{i,m} (\mathbf{J})_{m,j},$$

where $(\mathbf{PJ})_{i,j}$ is the (i,j)-th element of $\mathbf{PJ}$, and the i-th row sum becomes

$$\begin{aligned} \sum_j (\mathbf{PJ})_{i,j} &= \sum_j \sum_{m=1}^{n} (\mathbf{P})_{i,m} (\mathbf{J})_{m,j}, \\ &= \sum_j \Big((\mathbf{P})_{i,1} (\mathbf{J})_{1,j} + \ldots + (\mathbf{P})_{i,n} (\mathbf{J})_{n,j} \Big), \\ &= (\mathbf{P})_{i,1} \underbrace{\sum_j (\mathbf{J})_{1,j}}_{\leq 1} + \ldots + (\mathbf{P})_{i,n} \underbrace{\sum_j (\mathbf{J})_{n,j}}_{\leq 1}. \end{aligned} \tag{2.16}$$

Thus, we have

$$\sum_j (\mathbf{PJ})_{i,j} \leq (\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,n}. \tag{2.17}$$

Let us first consider *case (i)*, where the i-th row of $\mathbf{P}$ is sub-stochastic. From Eq. (2.17) and Assumption **A2**, we have

$$\sum_j (\mathbf{PJ})_{i,j} \leq \sum_{j=1}^{n} (\mathbf{P})_{i,j} \leq \beta_2 < 1. \tag{2.18}$$

Therefore, the i-th row becomes sub-stochastic after a sub-stochastic update at row i. We now consider *case (ii)* where the i-th row of $\mathbf{P}$ is stochastic, i.e.,

$$\sum_{m=1}^{n} (\mathbf{P})_{i,m} = 1.$$

In this case, $\sum_j (\mathbf{PJ})_{i,j}$ is a linear-convex combination of the row sums of $\mathbf{J}$, see Eq. (2.16), which is strictly less than one, if and only if $\mathbf{J}$ has at least one sub-stochastic row, say m', such that $(\mathbf{P})_{i,m'} \neq 0$, i.e.

$$\sum_j (\mathbf{PJ})_{i,j} = \underbrace{(\mathbf{P})_{i,m'}}_{\neq 0} \underbrace{(\mathbf{J})_{m',j}}_{<1} + \sum_{m \neq m'} (\mathbf{P})_{i,m} \underbrace{(\mathbf{J})_{m,j}}_{\leq 1}. \tag{2.19}$$

Therefore, $\sum_j (\mathbf{PJ})_{i,j} < 1$ and the lemma follows. □

Lemma 3. *With Assumptions **A0-A2**, a sub-stochastic row remains sub-stochastic throughout a slice.*

Proof. We use the notations of Lemma 2 on $\mathbf{J}$, $\mathbf{P}$, and Eq. (2.15), and rewrite Eq. (2.16) as

$$\begin{aligned} \sum_j (\mathbf{PJ})_{i,j} &= \sum_j \sum_{m=1}^{n} (\mathbf{P})_{i,m} (\mathbf{J})_{m,j}, \\ &= \sum_{m \in \mathcal{I}} \left((\mathbf{P})_{i,m} \sum_j (\mathbf{J})_{m,j} \right) \\ &+ \sum_{m \in \mathcal{U}} \left((\mathbf{P})_{i,m} \sum_j (\mathbf{J})_{m,j} \right). \end{aligned} \tag{2.20}$$

Let us consider the general case after the first success, where there exist $r \geq 1$ sub-stochastic rows in $\mathbf{J}$, i.e. $|\mathcal{I}| = r$, and $|\mathcal{U}| = n - r$. Without loss of

generality, suppose the r sub-stochastic rows of $\mathbf{J}$ lie in the first r rows. We need to show that if the i-th row in $\mathbf{J}$ is sub-stochastic, i.e. $i \leq r$, it remains sub-stochastic after a multiplication by either a stochastic or a sub-stochastic system matrix, $\mathbf{P}$. Rewrite the i-th row sum as

$$\begin{aligned}\sum_j (\mathbf{PJ})_{i,j} = (\mathbf{P})_{i,1} \sum_j (\mathbf{J})_{1,j} + \ldots + (\mathbf{P})_{i,r} \sum_j (\mathbf{J})_{r,j} \\ + (\mathbf{P})_{i,r+1} \underbrace{\sum_j (\mathbf{J})_{r+1,j}}_{=1} + \ldots + (\mathbf{P})_{i,n} \underbrace{\sum_j (\mathbf{J})_{n,j}}_{=1}.\end{aligned}$$

Thus,

$$\begin{aligned}\sum_j (\mathbf{PJ})_{i,j} &= (\mathbf{P})_{i,1} \sum_j (\mathbf{J})_{1,j} + \ldots + (\mathbf{P})_{i,r} \sum_j (\mathbf{J})_{r,j} \\ &+ (\mathbf{P})_{i,r+1} + \ldots + (\mathbf{P})_{i,n}, \end{aligned} \tag{2.21}$$

where we used the fact that in $\mathbf{J}$, any row $\in \mathcal{U}$ is stochastic.

Let us first consider the i-th row of $\mathbf{P}$ to be stochastic:

$$(\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r} + (\mathbf{P})_{i,r+1} + \ldots + (\mathbf{P})_{i,n} = 1.$$

Thus,

$$(\mathbf{P})_{i,r+1} + \ldots + (\mathbf{P})_{i,n} = 1 - \Big((\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r} \Big).$$

Therefore, from Eq. (2.21) we can write

$$\sum_j (\mathbf{PJ})_{i,j} = (\mathbf{P})_{i,1} \sum_j (\mathbf{J})_{1,j} + \ldots + (\mathbf{P})_{i,r} \sum_j (\mathbf{J})_{r,j}$$

$$+ \quad 1 - \Big((\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r} \Big). \tag{2.22}$$

Finally,

$$\begin{aligned} 0 \leq \sum_j (\mathbf{PJ})_{i,j} = \quad & 1 + \quad (\mathbf{P})_{i,1} \left(\sum_j (\mathbf{J})_{1,j} - 1 \right) \\ & \vdots \\ & + \quad (\mathbf{P})_{i,r} \left(\sum_j (\mathbf{J})_{r,j} - 1 \right), \\ \leq \quad & 1 + \quad (\mathbf{P})_{i,i} \left(\sum_j (\mathbf{J})_{i,j} - 1 \right), \end{aligned} \tag{2.23}$$

because the first r rows in $\mathbf{J}$ are sub-stochastic leading to $\sum_j (\mathbf{J})_{m,j} - 1 < 0$, for any $m = 1, \ldots, i, \ldots r$, and since the i-th row in $\mathbf{P}$ is stochastic, by Assumption **A1** we have

$$0 < \beta_1 \leq (\mathbf{P})_{i,i}.$$

Note that in Eq. (2.23) the only way to lose sub-stochasticity is to have $(\mathbf{P})_{i,m} = 0$ for all $m \leq r$. However, sub-stochasticity can be preserved by putting a non-zero weight on any row in $\mathcal{I}$. Since this knowledge in not available in general, a sufficient condition to ensure this is $0 < \beta_1 \leq (\mathbf{P})_{i,i}$. Thus, the i-th row sum remains strictly less than one (and greater than zero) after any stochastic update at the i-th row as long as Assumption **A1** is satisfied. Note that the lower bound on the j-th row sum stems from the non-negativity of system matrices.

Now consider the i-th row of $\mathbf{P}$ to be sub-stochastic. From **A2**, we have

$$(\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r} + (\mathbf{P})_{i,r+1} + \ldots + (\mathbf{P})_{i,n} \leq \beta_2 < 1.$$

Therefore,

$$(\mathbf{P})_{i,r+1} + \ldots + (\mathbf{P})_{i,n} \leq \beta_2 - \Big((\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r}\Big).$$

Thus, from Eq. (2.21) we can write

$$\begin{aligned} \sum_j (\mathbf{PJ})_{i,j} &\leq (\mathbf{P})_{i,1} \sum_j (\mathbf{J})_{1,j} + \ldots + (\mathbf{P})_{i,r} \sum_j (\mathbf{J})_{r,j} \\ &+ \beta_2 - \Big((\mathbf{P})_{i,1} + \ldots + (\mathbf{P})_{i,r}\Big). \end{aligned} \tag{2.24}$$

Finally,

$$\begin{aligned} 0 \leq \sum_j (\mathbf{PJ})_{i,j} \leq \beta_2 &+ (\mathbf{P})_{i,1} \left(\sum_j (\mathbf{J})_{1,j} - 1 \right) \\ &\vdots \\ &+ (\mathbf{P})_{i,r} \left(\sum_j (\mathbf{J})_{r,j} - 1 \right), \\ \leq \beta_2 &< 1, \end{aligned} \tag{2.25}$$

where again we used the fact that

$$\sum_j (\mathbf{J})_{m,j} - 1 < 0, \qquad m = 1, \ldots, i, \ldots r.$$

Eq.(2.25) shows that in case of a sub-stochastic i-th row in $\mathbf{P}_{k+1}$, this row remains sub-stochastic in $\mathbf{J}_{k+1}$, as long as Assumption **A2** is satisfied and the

conditions on individual weights are not required. Note the strict inequality, i.e., if $(\mathbf{P})_{i,m} \neq 0$ for any $m = 1, \ldots, r$, then

$$\sum_j (\mathbf{PJ})_{i,j} < \beta_2.$$

This lemma establishes that under the Assumptions **A0-A2**, sub-stochasticity is always preserved. □

These results describe the behavior of the sub-stochastic rows in the slices explicitly derived under the Assumptions **A0-A2**. The next results characterize the infinity norm bound on the slices. To this end, let us define $\beta_3(j)$, as the *maximum row sum over the sub-stochastic rows* of the product of all system matrices before $\mathbf{P}_j$ in the slice $\mathbf{M}$. Mathematically,

$$\beta_3(j) = \max_{m \in \mathcal{I}_{j-1}} \{v_m\}, \tag{2.26}$$

where v_m is the m-th element of

$$\mathbf{v}_{j-1} = (\mathbf{J}_{j-1})\mathbf{1}_n = (\mathbf{P}_{j-1} \ldots \mathbf{P}_3\mathbf{P}_2\mathbf{P}_1)\mathbf{1}_n,$$

and $\mathbf{1}_n$ is the column vector of n ones. The following lemma considers the case where no sub-stochastic update occurs at row i throughout a slice.

Lemma 4. *Assume no sub-stochastic update at the i-th row in a given slice, $\boldsymbol{M}$. With Assumptions **A0-A2**, the i-th row sum of this slice is upper bounded by*

$$1 + {\beta_1}^{|\boldsymbol{M}|-h_i+1}(\beta_3(h_i) - 1),$$

where the first success at row i occurs in the h_i-th update of this slice.

Proof. Eq. (2.23) expresses the i-th row sum after a stochastic update at row i. Clearly, before the first success, at index h_i, we have

$$\sum_j (\mathbf{J}_k)_{i,j} = 1, \qquad \forall k < h_i.$$

In order to find the maximum possible row sum for the i-th row at the end of a slice, we should find a scenario, which maximizes the row sum after the first success at index h_i and keeps maximizing it at each subsequent update. Consider Eq. (2.23) after the first success at index h_i. Since sub-stochastic updates are not allowed at row i from the lemma's statement, the first success can only occur via a stochastic update at the i-th row, and Assumption **A1** is applicable. Note that the right hand side of Eq. (2.23) is maximized if the minimum number of $(\mathbf{P})_{i,m\in\mathcal{I}}$'s are any non-zero. This is because each of such weights decreases the row sum. Now suppose $(\mathbf{P}_{h_i})_{i,r'}$ is the only non-zero weight among all $(\mathbf{P}_{h_i})_{i,j\in\mathcal{I}}$'s. In this case, Eq. (2.23) reduces to the following

$$\sum_j (\mathbf{P}_{h_i}\mathbf{J}_{h_i-1})_{i,j} = 1 - (\mathbf{P}_{h_i})_{i,r'}\left(1 - \sum_j (\mathbf{J}_{h_i-1})_{r',j}\right), \tag{2.27}$$

in which $r' \in \mathcal{I}_{h_i-1}$. Also note that $r' \neq i$, since row i is stochastic before the time instant, h_i. In order to maximize the right hand size of Eq. (2.27), $(\mathbf{P}_{h_i})_{i,r'}$ should be minimized, and $\sum_j (\mathbf{J}_{h_i-1})_{r',j}$ should be maximized. From Eq. (2.26), the maximum value of $\sum_j (\mathbf{J}_{h_i-1})_{r',j}$ before the first success is $\beta_3(h_i)$. Thus,

after the h_i-th update, where row i becomes sub-stochastic for the first time, we may write

$$\sum_j (\mathbf{P}_{h_i}\mathbf{J}_{h_i-1})_{i,j} \leq 1 - \beta_1 \left(1 - \beta_3(h_i)\right). \tag{2.28}$$

After this update, $\mathbf{J}_{h_i} = \mathbf{P}_{h_i} \ldots \mathbf{P}_2\mathbf{P}_1$, and

$$\beta_3(h_i) \leq \beta_3(h_i + 1) \leq 1 - \beta_1 \left(1 - \beta_3(h_i)\right), \tag{2.29}$$

where $\beta_3(h_i)$ is the r'-th row sum in $\mathbf{J}_{h_i-1}$, and $\beta_3(h_i + 1)$ is the i-th row sum in $\mathbf{J}_{h_i}$. Under this scenario, after the first success, the i-th row has the maximum sum over all rows of $\mathbf{J}_{h_i}$, and in order to increase this sum at the next update, the i-th row has to update only with itself. After the success at index h_i, the i-th row sum becomes less than one, and $r' = i$, for any subsequent update until the end of a slice. After the next update, $\mathbf{P}_{h_i+1}$, using the same argument we obtain

$$\begin{aligned}
\sum_j (\mathbf{P}_{h_i+1}\mathbf{J}_{h_i})_{i,j} &= 1 - (\mathbf{P}_{h_i+1})_{i,i} \left(1 - \sum_j (\mathbf{J}_{h_i})_{i,j}\right), \\
&\leq 1 - \beta_1 \left(1 - (1 - \beta_1(1 - \beta_3(h_i)))\right) \\
&= 1 - {\beta_1}^2 \left(1 - \beta_3(h_i)\right).
\end{aligned} \tag{2.30}$$

If row i keeps updating with itself, at the end of slice, after $|\mathbf{M}| - h_i$ number of such updates we have

$$\sum_j (\mathbf{P}_{|\mathbf{M}|}\mathbf{J}_{|\mathbf{M}|-1})_{i,j} \leq 1 + {\beta_1}^{|\mathbf{M}|-h_i+1}(\beta_3(h_i) - 1), \tag{2.31}$$

and the lemma follows. □

Next, we consider sub-stochastic updates at row i.

Lemma 5. *Assume at least one sub-stochastic update at the i-th row in slice, $\boldsymbol{M}$. With Assumptions **A0-A2**, the i-th row sum in $\boldsymbol{M}$ is upper bounded by*

$$1 + \beta_1{}^{|\boldsymbol{M}|-g_i}(\beta_2 - 1),$$

where the last sub-stochastic update at row i occurs in the g_i-th update in $\boldsymbol{M}$.

Proof. As shown in Eq. (2.25) and by Assumption **A2**, any sub-stochastic update at row i imposes the upper bound of β_2 on the i-th row sum. Thus, after the *last* sub-stochastic update at row i we have

$$\sum_j (\mathbf{P}_{g_i}\mathbf{J}_{g_i-1})_{i,j} \leq \beta_2 < 1.$$

After $\mathbf{P}_{g_i}$, there is no sub-stochastic update, and by Assumption **A1**, the i-th self-weight will be non-zero until the end of the slice. Following the same argument as in Lemma 4, the upper bound on the i-th row sum is maximized after each update if the i-th row does not update with any sub-stochastic row other than itself. For any update after the last success, Eq. (2.9) holds and we have:

$$\mathcal{N}_k^i \cap \mathcal{I}_k = \emptyset, \qquad \forall k > g_i. \tag{2.32}$$

After the $\mathbf{P}_{g_i+1}$-th update we have

$$\sum_j (\mathbf{P}_{g_i+1}\mathbf{J}_{g_i})_{i,j} \leq 1 - \beta_1\,(1 - \beta_2)\,, \tag{2.33}$$

and at the end of a slice, we have

$$\sum_j (\mathbf{P}_{|\mathbf{M}|}\mathbf{J}_{|\mathbf{M}|-1})_{i,j} \leq 1 + {\beta_1}^{|\mathbf{M}|-g_i}(\beta_2 - 1), \tag{2.34}$$

and the lemma follows. □

The following lemma combines Lemma 4 and Lemma 5 and relate them to the infinity norm bound of a slice.

Lemma 6. *With Assumptions **A0-A2**, for a given slice, **M**,*

$$\|\boldsymbol{M}\|_\infty \leq \max_i \{1 + {\beta_1}^{|\boldsymbol{M}|-l_i}(\beta - 1)\}, \tag{2.35}$$

where: $l_i = h_i - 1,\ \beta = \beta_3(h_i)$, when stochastic updates occur at row i; and, $l_i = g_i,\ \beta = \beta_2$, when sub-stochastic updates occur at row i.

The next lemma considers the worst-case scenario for the infinity norm of a slice, which provides an upper bound for Eq. (2.35).

Lemma 7. *With assumptions **A0-A2**, for the j-th slice we get*

$$\|\boldsymbol{M}_j\|_\infty \leq 1 - \alpha_j < 1, \qquad j \geq 0, \tag{2.36}$$

where

$$\alpha_j = f(|\boldsymbol{M}_j|, \beta_1, \beta_2) = {\beta_1}^{|\boldsymbol{M}_j|-1}(1 - \beta_2). \tag{2.37}$$

Proof. In order to find the maximum upper bound on the infinity norm of a slice, we consider a *worst case scenario*, in which a row sum incurs the largest

increase throughout the slice. To do so, we examine the maximum possible upper bound on the i-th row sum for the two cases discussed in Lemma 4 and Lemma 5 separately.

Consider no sub-stochastic update at the i-th row. We should find a scenario that maximizes the right hand side of Eq. (2.31). In addition, we need to make sure that such scenario is practical, i.e. all other rows become sub-stochastic before a slice is terminated. Since there are no sub-stochastic updates at row i, a slice can not be initiated by an update in row i, i.e. $h_i \geq 2$. At the initiation of a slice, one row other than i, becomes sub-stochastic, and the upper bound imposed on this row is β_2 by Assumption **A2**, hence $\beta_4(h_i) = \beta_4(2) = \beta_2$. Therefore, following the discussion in Lemma 4,

$$1 + {\beta_1}^{|\mathbf{M}_j|-1}(\beta_2 - 1), \tag{2.38}$$

provides the largest upper bound on the i-th row sum of $\mathbf{M}_j$. Note that this bound is feasible if we consider the following scenario: After row i becomes sub-stochastic at $h_i = 2$, we let the next $n-2$ updates for the other stochastic rows to become sub-stochastic, each updating *only* with the sub-stochastic row with the largest row sum. Thus, the largest row sum keeps increasing in the same manner as discussed in Lemma 4 within the next $n-2$ updates. At $(n+1)$-th update, row i again updates with a row, which has the maximum row sum in $\mathbf{J}_n$, and keeps updating by itself until the slice is terminated. The

aforementioned scenario is equivalent to the one, where the first success at row i occurs at $h_i = n$, and all other rows become sub-stochastic within the first $n-1$ updates, and

$$\beta_4(h_i = n) = 1 + {\beta_1}^{n-2}(\beta_2 - 1).$$

Now consider sub-stochastic updates at row i. The right hand side of Eq. (2.34) is maximized if g_i is minimized. In this case, the minimum value for g_i corresponds to a scenario where a sub-stochastic update at row i initiates a slice and no other sub-stochastic update occurs at this row. Using the same argument as before, all other rows become sub-stochastic within the next $n-1$ updates and the largest upper bound on the i-th row in this case is the same as the one given in Eq. (2.38). □

Finally, note that

$$\|\mathbf{M}\|_\infty \leq 1 + {\beta_1}^{|\mathbf{M}|-1}(\beta_2 - 1), \tag{2.39}$$

is the *largest* upper bound on the infinity norm of a given slice, $\mathbf{M}$.

2.4 Stability of discrete-time LTV systems

In this section, we study the stability of discrete-time, LTV dynamics with (sub-) stochastic system matrices. Recall that we are interested in the asymptotic stability of Eq. (2.3), such that the steady-state forgets the initial con-

ditions and is a function of inputs. A sufficient condition towards this aim is the convergence of the following to zero, for any $\mathbf{x}_0$:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{P}_k \mathbf{x}_k \\ &= \mathbf{P}_k \mathbf{P}_{k-1} \dots \mathbf{P}_0 \mathbf{x}_0, \end{aligned} \tag{2.40}$$

which is equivalent to

$$\lim_{k\to\infty} \mathbf{P}_k \mathbf{P}_{k-1} \dots \mathbf{P}_0 = \mathbf{0}_{n\times n},$$

where the subscript below $\mathbf{0}$ denotes its dimensions. Taking advantage of the slices, we study the following dynamics:

$$\mathbf{y}_{t+1} = \mathbf{M}_t \mathbf{y}_t, \qquad t \geq 0, \tag{2.41}$$

instead of Eq. (2.40), where $\mathbf{y}_0 = \mathbf{x}_0$, $\mathbf{y}_t = \mathbf{x}_{t_1}, t \geq 1$, with

$$t_1 = \sum_{i=1}^{t} |\mathbf{M}_i|.$$

Thus, for the convergence of the solution of Eq. (2.41) to zero, for any $\mathbf{y}_0$, we require

$$\lim_{t\to\infty} \mathbf{y}_{t+1} = \lim_{t\to\infty} \mathbf{M}_t \mathbf{M}_{t-1} \dots \mathbf{M}_0 \mathbf{y}_0 = \mathbf{0}_n.$$

We now provide our main result of this chapter in the following theorem:

Theorem 1. *With Assumptions **A0-A2**, the LTV system in Eq.* (2.41) *is absolutely asymptotically stable if any one of the following holds:*

(i) Each slice has a bounded length, i.e.

$$|\boldsymbol{M}_j| \leq N < \infty, \qquad \forall j,\ N \in \mathbb{N}; \tag{2.42}$$

(ii) There exists a set, $\boldsymbol{J}_1$, consisting of an infinite number of slices such that

$$|\boldsymbol{M}_j| \leq N_1 < \infty, \qquad \forall \boldsymbol{M}_j \in \boldsymbol{J}_1,\ \textit{and}\ |\boldsymbol{M}_j| < \infty, \forall \boldsymbol{M}_j \notin \boldsymbol{J}_1; \tag{2.43}$$

(iii) For every $i \in \mathbb{N}$, there exists a set, $\boldsymbol{J}_2$, of slices such that

$$\exists \boldsymbol{M}_j \in J_2 : \quad |\boldsymbol{M}_j| \leq \frac{1}{\ln(\beta_1)} \ln\left(\frac{1 - e^{(-\gamma_2 i^{-\gamma_1})}}{1 - \beta_2}\right) + 1, \tag{2.44}$$

for some $\gamma_1 \in [0, 1]$, $\gamma_2 > 0$, and $|\boldsymbol{M}_j| < \infty, j \notin J_2$.

Proof. Using the sub-multiplicative norm property in Eq. (2.40), we get

$$\|\mathbf{y}_{t+1}\|_\infty \leq \|\mathbf{M}_t\|_\infty \ldots \|\mathbf{M}_0\|_\infty \|\mathbf{y}_0\|_\infty. \tag{2.45}$$

Case (i): From Eqs. (2.36), (2.37) and (2.41), we have

$$\|\mathbf{M}_j\|_\infty \leq \delta < 1, \qquad \forall j, \tag{2.46}$$

where $\delta = 1 + {\beta_1}^{N-1}(\beta_2 - 1) < 1$, and this case follows.

Case (ii): We first note that the infinity norm of each slice has a trivial upper bound of 1. From Eq. (2.45), we have

$$\lim_{t\to\infty} \|\mathbf{y}_{t+1}\|_\infty \leq \lim_{t\to\infty} \prod_{j\in J_1} \|\mathbf{M}_j\|_\infty \prod_{j\notin J_1} \|\mathbf{M}_j\|_\infty \|\mathbf{y}_0\|_\infty,$$

$$\leq \lim_{t\to\infty} \prod_{j\in J_1} \|\mathbf{M}_j\| \|\mathbf{y}_0\|_\infty. \tag{2.47}$$

Similar to Case (i), this case follows by defining

$$\|\mathbf{M}_j\|_\infty \leq \delta_1 = 1 + {\beta_1}^{N_1-1}(\beta_2 - 1) < 1.$$

Case (iii): According to Lemma 7, Eq. (2.45) leads to

$$\lim_{t\to\infty} \|\mathbf{y}_{t+1}\|_\infty \leq \lim_{t\to\infty} \prod_{j=0}^{t} (1-\alpha_j) \|\mathbf{y}_0\|_\infty. \tag{2.48}$$

Consider the asymptotic convergence of the infinite product of a sequence $1-\alpha_j$ to 0. It can be verified that

$$\lim_{t\to\infty} \prod_{j=1}^{t} (1-\alpha_j) = 0, \text{ or } \lim_{t\to\infty} \sum_{j=1}^{t} (-\ln(1-\alpha_j)) = \infty. \tag{2.49}$$

Now note that

$$\sum_{i=1}^{\infty} \gamma_2 i^{-\gamma_1} = \infty, \qquad \text{for } 0 \leq \gamma_1 \leq 1, 0 < \gamma_2,$$

because $\frac{1}{i^{\gamma_1}}$ sums to infinity for all values of γ_1 in $[0, 1]$, and multiplying by a positive number, γ_2, does not change the infinite sum. It can be verified that Eq. (2.49) holds when

$$-\ln(1-\alpha_j) \geq \gamma_2 i^{-\gamma_1},$$

subsequently resulting into:

$$1 - \alpha_j \leq e^{(-\gamma_2 i^{-\gamma_1})},$$

for some $\gamma_1 \in [0, 1]$, and $0 < \gamma_2$. Therefore if for any $i \in \mathbb{N}$, there exists a slice, $\mathbf{M}_j$, in $\mathbf{J}_2$ such that

$$\|\mathbf{M}_j\|_\infty \leq 1 - \alpha_j \leq e^{(-\gamma_2 i^{-\gamma_1})}, \tag{2.50}$$

we get

$$\lim_{t\to\infty} \prod_{j=0}^{t} (1 - \alpha_j) = \prod_{j\in J_2} (1 - \alpha_j) \prod_{j\notin J_2} (1 - \alpha_j) = 0, \tag{2.51}$$

and the asymptotic stability follows. By substituting α_j from Lemma 7 in the left hand side of Eq. (2.50), we get

$$1 - {\beta_1}^{|\mathbf{M}_j|-1}(1 - \beta_2) \leq e^{(-\gamma_2 i^{-\gamma_1})},$$

which leads to

$$\ln\left(\frac{1 - e^{(-\gamma_2 i^{-\gamma_1})}}{1 - \beta_2}\right) \quad \leq \quad (|\mathbf{M}_j| - 1) \ln \beta_1. \tag{2.52}$$

Since $\beta_1 < 1$, $\ln \beta_1$ is negative and dividing both sides of Eq. (2.52) by a negative number changes the direction of the inequality, i.e.,

$$|\mathbf{M}_j| \leq \frac{1}{\ln(\beta_1)} \ln\left(\frac{1 - e^{(-\gamma_2 i^{-\gamma_1})}}{1 - \beta_2}\right) + 1. \tag{2.53}$$

Now note that in the right hand side of Eq. (2.53), the first ln is negative; for the bound to remain meaningful, the second ln must also be negative that requires $\beta_2 < e^{(-\gamma_2 i^{-\gamma_1})}$, which holds for any value of $\beta_2 \in [0, 1)$ by choosing an appropriate $\gamma_2 > 0$.

To conclude, we note that if the slices are such that there exists a slice with length following Eq. (2.53) for every $i \in \mathbb{N}$, not necessarily in any order, the infinite product of such slices goes to a zero matrix. Finally, from Eq. (2.48)

$$\lim_{t\to\infty} \|\mathbf{y}_{t+1}\|_\infty = \mathbf{0},$$

which completes the proof in this case. □

Unbounded connectivity: In the following, we shed some light on Case (iii) and Eq. (2.53). First, note that Eq. (2.53) does not require the slice indices to be i. In other words, the slice lengths are not growing as i increases and slices satisfying Eq. (2.53) may appear in any order. For the next argument, note that the right hand side of Eq. (2.53) goes to $+\infty$ as $i \to \infty$; because $e^{(-\gamma_2 i^{-\gamma_1})}$ goes to 1. A longer slice length can be related to a slow information propagation in the network. Eq. (2.53) further shows that LTV stability does not require bounded slice lengths (as in cases (i) and (ii)); the convergence is guaranteed as long as a well-behaved sub-sequence of slices exist, whose lengths increase according to a certain exponential rate. Next note that $\gamma_1 = 1$ is valid and corresponds to the fastest growing exponential, $e^{(-\gamma_2 i^{-1})}$, whose infinite product is 0. This means that only a sub-sequence of slices need to behave such that their behavior is not worse than $e^{-\gamma_2 i^{-1}}$, in any order. We may write this requirement as

$$\mathbb{P}\left(\mathbf{M}_j \text{ exists for some } j \text{ such that } \|\mathbf{M}_j\|_\infty \leq e^{-\gamma_2 i^{-1}}\right) = 1,$$

$\forall i \geq 1$ and $0 < \gamma_2$, where $\mathbb{P}$ denotes the probability of the corresponding event. On the other hand, by choosing $\gamma_1 = 0$ the upper bound on the slice length in Case (iii) becomes a constant. Hence, the first two cases are special cases of this bound if we set N and N_1 as

$$\frac{1}{\ln(\beta_1)} \ln\left(\frac{1 - e^{(-\gamma_2)}}{1 - \beta_2}\right) + 1.$$

2.5 Simulations

In this section, we provide a simple yet insightful scenario that demonstrates the main contributions of this chapter in the light of Theorem 1. Consider a network with $n = 4$ agents and $s = 1$ anchor with the following update scenario: each slice, t, starts with an agent, say 1, becoming informed by updating with the anchor and an uninformed agent, say 2; next, an uninformed agent, say 2, becomes informed by updating with agents, say 1 and 3; next, an informed agent, say 2 updates with uninformed agents, say 3 and 4, for a total of n_t iterations; finally, the remaining uninformed agents become informed in two successive iterations by updating with the informed agents[6] (and/or anchor). Note that $|\mathbf{M}_t| = 2 + n_t + 2$.

To show the applicability of Theorem 1, we choose n_t as the right hand side of Eq. (2.53) with $\beta_1 = 0.99, \beta_2 = 0.4, \gamma_1 = 1, \gamma_2 = 0.7$. The slice

[6]In this simulation, the weights are randomly assigned to the neighboring nodes such that Assumptions **A0-A2** are satisfied.

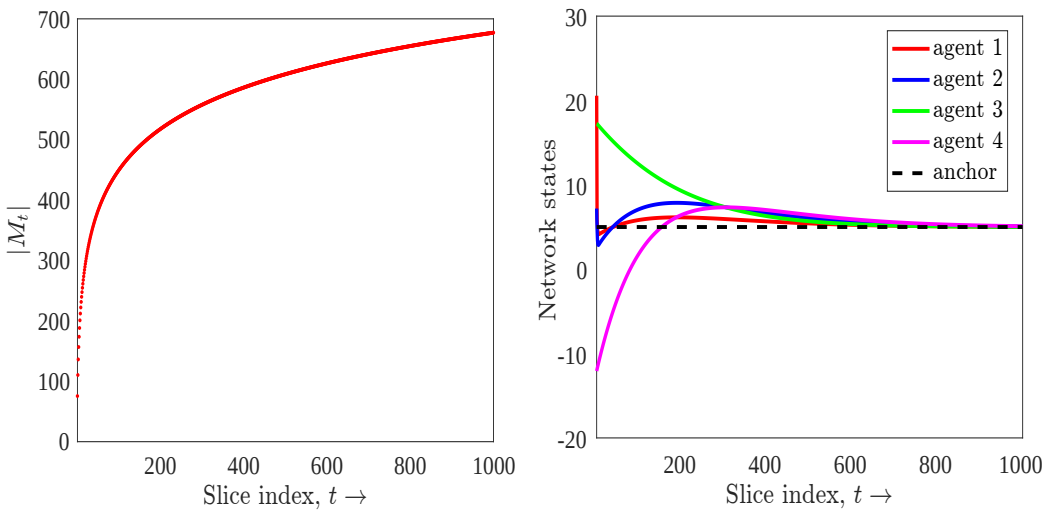

Figure 2.2: (Left) Unbounded slice lengths. (Right) Convergence of the agents' states to the leader state.

length, n_t+4, with these parameters is shown in Fig. 2.2 (Left). It is clear that the slice lengths do not have a uniform bound and, in fact, remain finite and unbounded due to which the existing results are not applicable as they assume a uniform bound on n_t. However, as shown in Theorem 1, the slice lengths (the rate at which the information disseminates from the anchor to the agents) are not required to have a uniform bound as long as the growth follows Eq. (2.53) for some γ_1, γ_2, see Fig. 2.2 (Left). We show the convergence results in Fig. 2.2 (Right). We emphasize that the updating scenario and the generation of slice lengths chosen in this experiment are without loss of generality.

Finally, we note that every growth of the slice lengths does not necessarily

ensure convergence. To show this, we note that

$$\lim_{|\mathbf{M}_j|\to\infty} \|\mathbf{M}_j\|_\infty = 1,$$

from Eq. (2.35) in Lemma 3. Thus, there exists a slice, $\mathbf{M}_j$, possibly with an arbitrarily large length, such that

$$\epsilon \leq \|\mathbf{M}_j\|_\infty \leq 1,$$

for any $\epsilon > 0$. For example, if there exists slices with lengths such that

$$\frac{a_j}{2} \leq \|\mathbf{M}_j\|_\infty, \qquad a_j = \sqrt{2 + a_{j-1}}, a_1 = \sqrt{2},$$

then,

$$\frac{2}{\pi} = \prod_j \frac{a_j}{2} \leq \prod_j \|\mathbf{M}_j\|_\infty \leq 1,$$

see Viéte's formula, and the slice product does not go to zero.

2.6 Summary

In this chapter, we study asymptotic stability of linear time-varying systems with randomly chosen, stochastic or sub-stochastic system matrices. Instead of exploring the joint spectral radius of the (infinite) set of system matrices, we partition them into non-overlapping slices, such that each slice has a subunit infinity norm, and the slices cover the entire sequence of the system matrices.

We use the infinity norm to characterize the asymptotic stability and provide upper bounds on the infinity norm of each slice as a function of its length. We show that asymptotic stability is guaranteed not only in the trivial cases where all (or an infinite subset) of the slices have a bounded length, but also if there exists an infinite subset of slices whose (unbounded) lengths do not grow faster than a particular exponential growth.

Chapter 3

Dynamic leader-follower

In this chapter, we study the consensus-based leader-follower algorithm in mobile multi-agent networks, where the goal for the entire network is to converge to the state of a leader[1]. We capture the mobility in the leader-follower algorithm by abstracting it as a linear time-varying system with random system matrices. In particular, a mobile agent, moving randomly in a bounded region, updates its state when it finds neighbors; and does not update when it does not find any node within its communication range. In this context, we develop certain regularity conditions on the system and input matrices such that each follower converges to the state of the leader. We use the approach described in Chapter 2 to analyze the corresponding LTV system; we parti-

[1]Note that in the context of multi-agent networks, we refer to *followers* and *leaders* as *agents* and *anchors*, respectively.

tion the entire chain of system matrices into non-overlapping slices, and relate the convergence of the multi-agent network to the lengths of these slices. In contrast to the existing results, we show that a bounded length on the slices, capturing the dissemination of information from the leader to the followers, is not required; as long as the slice-lengths are finite and do not grow faster than a certain exponential rate.

3.1 Introduction

A mobile multi-agent network is composed of a collection of mobile agents with possibly limited sensing, communications, and computation capabilities. Since mobility of the agents is becoming increasingly important in many practical applications, [78–81], e.g., monitoring a hazardous environment where a static network can not be deployed, mobile multi-agent networks have attracted a significant research attention in recent years.

In multi-agent networks, it is often desirable that the entire network reaches an agreement regarding a quantity of interest. This problem is referred to as *consensus*, which has applications in, e.g., hypothesis testing, [82–84], estimation, [85, 86], distributed optimization, [70], and diffusion, [87]; work on variants of consensus can be found in, [88–94]. In some applications, agents should be guided to a target state by introducing leaders, [56, 95]. Only the

leaders are aware of the common goal of the network, and the agents must converge to the state of the leader. The leader plays the role of an input whose influence is propagated throughout the network via a distributed algorithm, specifying the information exchange between an agent and its neighbors, [96]. Leader-follower networks have been widely used in many practical applications such as vehicle formation control, and robotic systems, [97, 98].

Related work on the conditions under which the states of the agents converge to the leader state can be found in, [65, 99–102]. In particular, Ref. [99] shows that the network converges to the state of the leader if there exists an infinite sequence of contiguous, non-overlapping and *bounded* time-intervals with the property that the union of the graphs across each interval is *strongly-connected*. Subsequently, Refs. [65, 100] extend the results in [99] to directed graphs, and prove that the coordination is possible if the union of the interaction graphs has a spanning tree *frequently enough*. Ref. [101] presents a graph-theoretic approach to study consensus in dynamically changing environments, where each graph can be represented by a stochastic matrix. Ref. [102] studies the convergence of the product of asymmetric stochastic matrices, and relates the convergence rate to the second largest eigenvalue of the product.

In this chapter, we consider a leader-follower problem where the agents move randomly in a bounded region of interest and have a limited communication range. We assume gossip-based communication, [103], which is widely

used, e.g., [104], due to its simplicity and robustness. In particular, we assume that only one agent at each iteration exchanges information with nearby nodes and updates its state. Since the motion is random, an agent may not find neighbors at all times and an arbitrary agent may not communicate with a leader at any given time. This leads to different update scenarios, which we discuss thoroughly in Section 3.2. We abstract this updating procedure by a linear time-varying system and develop the conditions under which this LTV system converges to the leader state regardless of the agents' initial conditions. In contrast to a majority of the existing works, which deal with non-negative, stochastic matrices, the system matrices in a dynamic leader-follower algorithm are both *stochastic* or *sub-stochastic*, due to which the results focusing on stochastic matrices alone are not applicable.

In order to develop the results for (sub-) stochastic matrices, we use the notion of non-overlapping slices–the smallest product of consecutive system matrices. Each slice has an infinity norm of less than one *and* slices cover the entire system matrices. One complete slice implies information propagation from the leader to every other follower in the network. Clearly, if this information dissemination is completed in a bounded time, i.e., the slices have a uniform bound on their lengths, as the information from the leader reaches the entire network infinitely often, the asymptotic behavior can be developed rather straightforwardly. The most important contribution of this chapter is

that the (dynamic) leader-follower algorithm converges even when the slice lengths are unbounded; what is required is that the slice lengths do not grow larger at a rate faster than a certain exponential growth. In other words, the information from the leader reaches the rest of the network in an unbounded number of steps, the rate of which is explicitly characterized.

We now describe the rest of this chapter. In Section 3.2, we formulate the problem and describe the dynamics of a leader-follower network. In Section 3.3, we provide the sufficient conditions for the convergence of the network to the state of the leader. We provide simulation results in Section 3.4 and finally, Section 3.5 concludes this chapter.

3.2 Problem formulation

Consider a network with n followers (agents) in the set Ω, and one leader (anchor). We assume that all of the nodes (leader and followers) move arbitrarily and the followers exchange information if they find nearby nodes. For the i-th follower, let $\mathcal{N}_k^i$ be the set of neighbors (not including follower i) at time k. We also define $\mathcal{D}_k^i = \mathcal{N}_k^i \cup \{i\}$.

Leader-follower dynamics: The leader's state is fixed, whereas the state of a follower is influenced by its neighbors[2]. Note that due to mobility, it is not

[2]Throughout this chapter, we assume that the state of each node (agent or anchor) is a scalar.

guaranteed for a follower to find a leader among its neighbors at any time k. In fact, at a given time, it is possible that a follower does not have any neighbor. Let $x_k^i \in \mathbb{R}$ be the state of the i-th follower at time k and let $u \in \mathbb{R}$ be the leader state. The updating follower, $i \in \Omega$, implements the following:

(i) Follower i keeps its state when it has no neighbor:

$$x_{k+1}^i = x_k^i. \tag{3.1}$$

(ii) With no leader among neighbors, the state-update is:

$$x_{k+1}^i = \sum_{j \in \mathcal{D}_k^i} p_k^{ij} x_k^j, \tag{3.2}$$

where p_k^{ij} is the weight the updating follower, i, assigns to follower j that is among its neighbors at time k.

(iii) With the leader among neighbors, the state-update is

$$x_{k+1}^i = \sum_{j \in \mathcal{D}_k^i \cap \Omega} p_k^{ij} x_k^j \; + \; b_k^i u, \tag{3.3}$$

where $u \in \mathbb{R}$ is the leader state and b_k^i is the weight assigned by the updating follower, i, to the leader state at time k.

The above update scenarios can be abstracted by the following LTV system:

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k u, \qquad k \geq 0, \tag{3.4}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state of the followers at time k; $\mathbf{P}_k = \{p_k^{ij}\}$ and $\mathbf{B}_k = \{b_k^i\}$ are the time-varying system and input matrices, respectively. Note that we consider a network with only one leader, hence, the fixed state of the leader, u, is a scalar. In this case, $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{B}_k \in \mathbb{R}^{n\times 1}$. In case of a network with multiple, say s, leaders, $\mathbf{u} \in \mathbb{R}^s$ is the input vector and $\mathbf{B}_k \in \mathbb{R}^{n\times s}$.

Assumptions: We make the following assumptions:

A0: The state-update is *linear-convex*, i.e.,

$$\sum_j p_k^{ij} + b_k^i = 1, \qquad \forall k, i \in \Omega. \tag{3.5}$$

A1: When there is no leader among the neighbors, then

$$0 < \beta_1 \leq p_k^{ij} < 1, \qquad \beta_1 \in \mathbb{R},\ \forall i \in \Omega,\ j \in \mathcal{D}_k^i. \tag{3.6}$$

A2: When a leader is involved in an update, it always contributes a certain amount of information, i.e.,

$$0 < \beta_2 \leq b_k^i < 1, \qquad \beta_2 \in \mathbb{R},\ \forall i \in \Omega. \tag{3.7}$$

Remarks: The above assumptions are realistic and rather common in the related literature. While stochasticity is standard in sensor fusion and relevant applications, see e.g., [56, 95, 105], Eq. (3.6) implies that when a state update occurs without a leader, a non-zero self-weight is always assigned to

the (updating) follower's current state. This assumption does not allow a follower to completely forget its past information. Note that Assumption **A1** is required to maintain sub-stochasticity of the process and is necessary for each follower that does not directly communicate with the leader. This is because this follower can lose information received (indirectly) from the leader by updating with the neighbors that do not have any such information. Next, Eq. (3.7) restricts the amount of (unreliable) information received from other neighboring followers by guaranteeing that a certain information is always contributed by the leader. We note that it is possible for multiple followers to update at the same time and the assumption that only one follower updates is without loss of generality. Note that random motion in a bounded region is equivalent to saying that each follower communicates intermittently with other nodes with a non-zero probability. If a subset of followers are isolated, and never communicates with the rest of the network, they will not follow the convergence.

Under the above assumptions the LTV system matrices, $\mathbf{P}_k$'s, are always non-negative, and may be *identity*-when no update occurs; *stochastic*-when there is no leader among the neighbors; and, *sub-stochastic*-when the neighbors include the leader. In the next section, we use the framework introduced in Chapter 2 to study the convergence of the leader-follower algorithm represented by Eq. (3.4), under Assumptions **A0-A2**.

3.3 Leader-follower convergence

In this section, we use the results on the convergence of an infinite product of (sub-) stochastic matrices, from Section 2.4, to study Eq. (3.4). We provide the main result of this chapter in the following theorem:

Theorem 2. *Consider n followers and one leader moving in a finite and bounded region with follower state updates given in Eqs.* (3.1)-(3.3)*. Suppose Assumptions* ***A0-A2*** *hold. Then, for any random (or deterministic) motion that satisfies Eq.* (2.44) *on the corresponding slices, the followers (asymptotically) converge to the leader state.*

Proof. The random motion of the followers in a finite, bounded region results in the following LTV system:

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k u, \qquad k \geq 0, \tag{3.8}$$

where u is the state of the leader, and $\mathbf{P}_k$ is random and may be either identity, stochastic, or sub-stochastic. Hence, with Assumptions **A0-A2**,

$$\prod_{k=1}^{\infty} \mathbf{P}_k = 0, \tag{3.9}$$

under the conditions stated in Theorem 1. What is left to show is that when Eq. (3.9) holds and the leader state is fixed, all agents converge to the leader state. Using the slice notation, we may use a new time index, t, and rewrite

Eq. (3.8) as

$$\mathbf{y}_{t+1} = \mathbf{M}_t \mathbf{y}_t + \mathbf{N}_t u, \tag{3.10}$$

such that $\mathbf{y}_0 = \mathbf{x}_0$ and

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{x}_k, \qquad k = \sum_{i=1}^{t} |\mathbf{M}_i|,\ t \geq 1, && (3.11)\\
\mathbf{M}_t &= \mathbf{P}_{\left(\sum_{i=0}^{t} |\mathbf{M}_i|\right)-1} \ldots \mathbf{P}_{\left(\sum_{i=0}^{t-1} |\mathbf{M}_i|\right)},\ t > 0, && (3.12)\\
\mathbf{N}_t &= \sum_{m=0}^{|\mathbf{M}_t|-1} \left(\prod_{j=1}^{m} \mathbf{P}_{|\mathbf{M}_t|-j} \right) \mathbf{B}_{|\mathbf{M}_t|-1-m}, && (3.13)
\end{aligned}$$

with

$$\mathbf{M}_0 = \mathbf{P}_{|\mathbf{M}_0|-1} \ldots \mathbf{P}_0.$$

For simplicity, let us represent the t-th slice in Eq. (3.12) as:

$$\mathbf{M}_t \triangleq \mathbf{P}_T \mathbf{P}_{T-1} \ldots \mathbf{P}_0, \qquad |\mathbf{M}_t| = T + 1,$$

such that

$$\left(\sum_{i=0}^{t} |\mathbf{M}_i| - 1 \right) - \left(\sum_{i=0}^{t-1} |\mathbf{M}_i| \right) = T + 1.$$

In addition, we can rewrite Eqs. (3.8) and (3.10) as

$$\begin{aligned}
\mathbf{x}_{k+1} &= (\mathbf{P}_k \ldots \mathbf{P}_0)\mathbf{x}_0 + \sum_{m=0}^{k} \left(\prod_{j=1}^{m} \mathbf{P}_{k-j+1} \right) \mathbf{B}_{k-m} u,\\
\mathbf{y}_{t+1} &= (\mathbf{M}_t \ldots \mathbf{M}_0)\mathbf{y}_0 + \sum_{m=0}^{t} \left(\prod_{j=1}^{m} \mathbf{M}_{t-j+1} \right) \mathbf{N}_{t-m} u,
\end{aligned}$$

respectively. In Eq. (3.10), $\mathbf{y}_{t+1}$ asymptotically converges to a limit, say $\mathbf{y}^*$, because u is a constant, and the spectral radius, $\rho(\mathbf{M}_t)$, of the t-th slice is

strictly less than one, under the conditions of Theorem 1, i.e.,

$$\rho(\mathbf{M}_t) \leq \|\mathbf{M}_t\|_\infty < 1, \qquad \forall t. \tag{3.14}$$

Thus,

$$\lim_{t\to\infty} \mathbf{y}_{t+1} = \mathbf{y}^*.$$

Note that for any given $t \geq 1$ we can find the corresponding k from Eq. (3.11), and we have

$$\mathbf{x}^* = \mathbf{y}^* = \mathbf{M}_t\mathbf{y}^* + \mathbf{N}_t u \quad \Rightarrow \quad (I_n - \mathbf{M}_t)\mathbf{y}^* = \mathbf{N}_t u, \tag{3.15}$$

in which $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. Therefore,

$$\mathbf{x}^* = (\mathbf{I}_n - \mathbf{M}_t)^{-1}\mathbf{N}_t u. \tag{3.16}$$

Note that $(\mathbf{I} - \mathbf{M}_t)$ is invertible due to Eq. (3.14).

In order to show that the limiting states of the followers are indeed the leader state, we need to show

$$(\mathbf{I}_n - \mathbf{M}_t)^{-1}\mathbf{N}_t = \mathbf{1}_n \quad \Rightarrow \quad \mathbf{M}_t\mathbf{1}_n + \mathbf{N}_t = \mathbf{1}_n. \tag{3.17}$$

Note that since there is only one leader in the network, $\mathbf{N}_t$ is a $n \times 1$ column vector. By substituting $\mathbf{M}_t$ and $\mathbf{N}_t$ from Eqs. (3.12) and (3.13) into Eq. (3.17), we need to show that

$$(\mathbf{P}_T \dots \mathbf{P}_0)\mathbf{1}_n + \sum_{m=0}^{T} \left(\prod_{j=1}^{m} \mathbf{P}_{T+1-j} \right) \mathbf{B}_{T-m} = \mathbf{1}_n. \tag{3.18}$$

By expanding the left hand side of the above, we have

$$\begin{aligned}(\mathbf{P}_T\mathbf{P}_{T-1}\dots\mathbf{P}_0)\mathbf{1}_n &+ (\mathbf{P}_T\mathbf{P}_{T-1}\dots\mathbf{P}_1)\mathbf{B}_0 \\ &+ (\mathbf{P}_T\mathbf{P}_{T-1}\dots\mathbf{P}_2)\mathbf{B}_1 \\ &\vdots \\ &+ (\mathbf{P}_T\mathbf{P}_{T-1})\mathbf{B}_{T-2} \\ &+ (\mathbf{P}_T)\mathbf{B}_{T-1} \\ &+ (\mathbf{B}_T). \end{aligned} \tag{3.19}$$

The first line of the above expression can be simplified as

$$(\mathbf{P}_T\mathbf{P}_{T-1}\dots\mathbf{P}_1)(\mathbf{P}_0\mathbf{1}_n + \mathbf{B}_0), \tag{3.20}$$

in which $\mathbf{B}_0 \neq 0$ is a $n \times 1$ vector corresponding to the first sub-stochastic update at the beginning of the slice, $\mathbf{M}_t$. Also, $\mathbf{B}_0$ has only one non-zero, say α_i, at the i-th position if follower i updates with the leader at the beginning of the slice, $\mathbf{M}_t$. From Eq. (3.5), which is natural in leader-follower settings, [106], it can be verified that

$$\mathbf{P}_0\mathbf{1}_n + \mathbf{B}_0 = \mathbf{1}_n. \tag{3.21}$$

Therefore (3.19) reduces to the following

$$(\mathbf{P}_T\mathbf{P}_{T-1}\dots\mathbf{P}_1)\mathbf{1}_n + (\mathbf{P}_T\dots P_2)\mathbf{B}_1 + \dots + (\mathbf{B}_T). \tag{3.22}$$

After the first update, $\mathbf{B}_j$'s, $(1 < j \leq T-1)$, are non-zeros in case of sub-stochastic updates, and zeros otherwise. The procedure continues in a similar way for any sub-stochastic update, i.e. update with the leader. Let us now consider the other case, where the update is stochastic. Suppose $\mathbf{B}_c$ is the next sub-stochastic update, and we have

$$\mathbf{B}_j = 0, \qquad 1 \leq j < c.$$

Then, (3.22) reduces to

$$(\mathbf{P}_T \dots \mathbf{P}_{c+1})(\mathbf{P}_c \mathbf{P}_{c-1} \dots \mathbf{P}_1 \mathbf{1}_n + \mathbf{B}_c) + \dots + (\mathbf{B}_T). \tag{3.23}$$

Since between $\mathbf{P}_1$ and $\mathbf{P}_c$ there is no sub-stochastic update, we have

$$\mathbf{P}_{c-1} \dots \mathbf{P}_1 \mathbf{1}_n = \mathbf{1}_n,$$

and we can rewrite (3.23) as

$$(\mathbf{P}_T \dots \mathbf{P}_{c+1})(\mathbf{P}_c \mathbf{1}_n + \mathbf{B}_c) + \dots + (\mathbf{B}_T), \tag{3.24}$$

and the procedure continues as before (note the similarity between (3.20), and the first term on the left hand side of (3.24)). Finally,

$$\begin{aligned} (\mathbf{P}_T \dots \mathbf{P}_0)\mathbf{1}_n &+ \sum_{m=0}^{M} (\prod_{j=1}^{m} \mathbf{P}_{T+1-j}) \mathbf{B}_{T-m} \\ &= \mathbf{P}_T \mathbf{1}_n + \mathbf{B}_T \\ &= \mathbf{1}_n, \end{aligned} \tag{3.25}$$

which validates Eq. (3.18) and leads to

$$\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^* = u.$$

□

3.4 Simulations

In this section, we provide an illustrative example to demonstrate the concepts described in this chapter. In Fig. 3.1, we show the dynamic leader-follower algorithm, with $n = 6$ mobile followers and one mobile leader. The goal for the entire network is to converge to a common value in state space, i.e., the state of the leader. We set the communication radius of all followers to $r = 1.5$, where the region of interest is a 20×20 square. Each node can only explore a restricted region shown as a shaded area where the motion is restricted. Fig. 3.1 shows the random trajectories of each node for the first 25 iterations, as well as all possible update scenarios. In this figure, red triangle indicates the leader, blue circles represent the followers and red circles show the communication radius of each node. In this setup, only two of the followers are able to communicate directly to the leader, while other followers never lie in the communication radius of the leader. The information from the leader can reach these followers (indirectly) only by propagation through the other followers. Fig. 3.1 (bottom left) shows the case when only one follower communicates

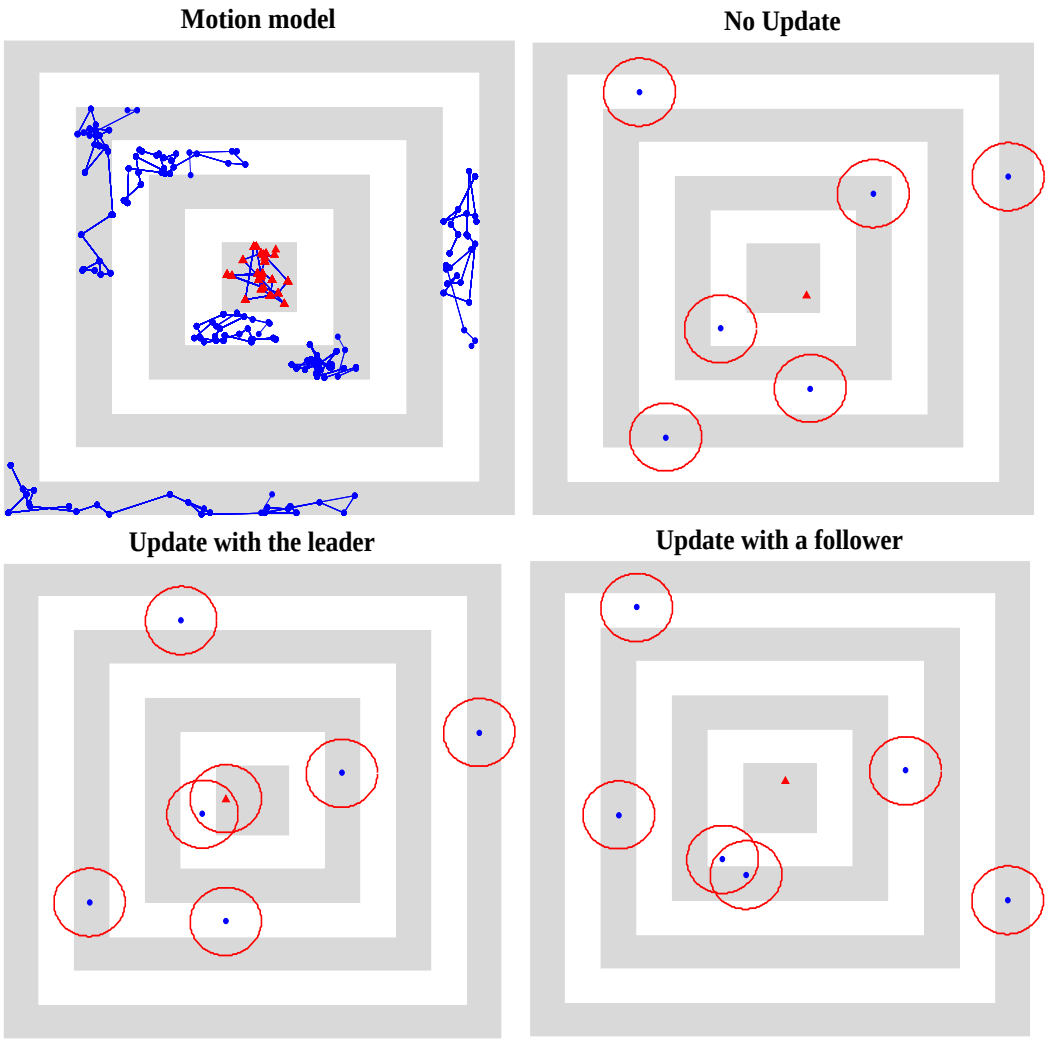

Figure 3.1: Updates and motion in a leader-follower network of size 7.

with the leader; resulting in a sub-stochastic system matrix, whereas Fig. 3.1 (bottom right) depicts the information exchange between two followers. This setup can be extended to arbitrary network configurations and sizes, where the communication and motion models ensure that the information reaches from the leader to each follower node. Finally, Fig. 3.2 illustrates the convergence

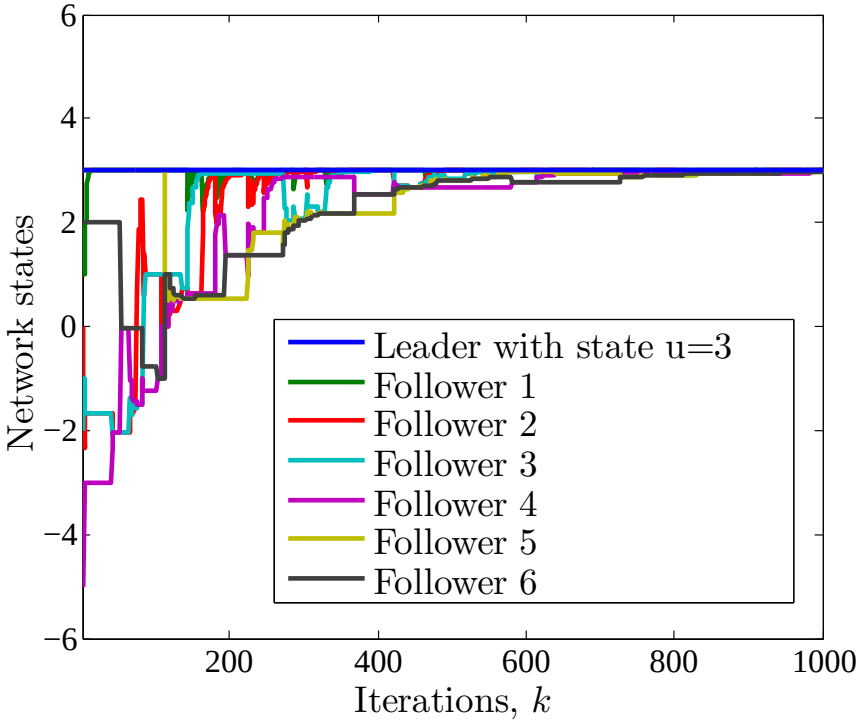

Figure 3.2: Convergence of 6 followers in a dynamic leader-follower network.

of follower states to the state of the leader, chosen at $u = 3$ and indicates by a blue line.

3.5 Summary

In this chapter, we study the leader-follower problem in mobile multi-agent networks, where the mobility of the agents (followers) result in dynamic updating scenarios. Abstracting such updates as an LTV system with randomly appearing stochastic or sub-stochastic system matrices, we establish the conditions under which the corresponding algorithm converges to the state of a

leader. In particular, we show that convergence is guaranteed if the motion of the followers and the leader follow a certain information propagation rate; and certain weights chosen by each follower have a uniform lower bound. Furthermore, the rate at which the information goes from the leader to each follower does not have to be bounded as long as it is finite and does not grow faster than a certain exponential growth. Although we focus on the networks with only one leader, the results can be easily generalized to any network with multiple leaders, where the followers converge to a linear-convex combination of the leader states.

Chapter 4

An opportunistic linear-convex algorithm for localization in dynamic networks

In order to navigate reliably and perform useful tasks in dynamic networks, a mobile agent must know its exact location. Localization, estimation of an agent's location from sensor data, is thus a fundamental problem in providing autonomous capabilities to a mobile agent. In this chapter, we develop a *distributed* algorithm to localize a network of agents moving arbitrarily in a bounded region. In the case of such mobile networks, the main challenge is that the agents may not be able to find nearby nodes to implement a distributed

algorithm. We address this issue by providing an *opportunistic* algorithm that only implements a location update when there are nearby nodes and does not update otherwise. We assume that each agent measures a noisy version of its motion and the distances to the nearby nodes. To localize a network of mobile agents in $\mathbb{R}^m$ ($m \geq 2$), we provide a simple *linear-convex* update, which is based on *barycentric coordinates*. We abstract the corresponding localization algorithm as an LTV system and using the results from Chapter 2, we show that it asymptotically converges to the true locations of the agents.

We first focus on the noiseless case, where the distance and motion vectors are known (measured) perfectly, and provide sufficient conditions on the convergence of the algorithm. We then evaluate the performance of the algorithm in the presence of noise and provide modifications to counter the undesirable effects of noise. We further show that our algorithm precisely tracks a mobile network as long the network includes one node with known location. This is in stark contrast to the requirement of at least $m + 1$ nodes with known location (in $\mathbb{R}^m$) that is prevalent in the localization literature. We show that the remaining degrees of freedom are, in fact, delivered by the motion.

4.1 Introduction

With fifth generation (5G) technologies looming in the horizon, there is a potential for networking vast numbers of heterogeneous devices, the Internet-of-Things (IoT), not only of static devices, but also of moving objects, users, or vehicles, [13–18, 107]. Location-awareness, providing the physical location of every static or moving object or agent, will enhance the ability to deploy new services and better management of the overall 5G system. Beyond these, location-aware technologies can also enable a variety of other applications from precision agriculture, [19], to intruder detection, [20], health care, [21], asset tracking, ocean data acquisition, [108], or emergency services, [22]. For example, location information is essential in providing an effective response in disasters such as fire rescue situations. Other relevant applications include military sensing, [23], physical security, industrial and manufacturing automation, and agentics, [109]. In addition, localization is essential in randomly deployed networks, where manual positioning of objects is not practical and the location of network nodes may change during run-time.

Although it is possible to include on-board Global Positioning System (GPS) receivers on each device to provide the position information, the use of GPS adds to the expense and power requirements and reduces the service life of Wireless Sensor Networks (WSNs). Moreover, GPS receivers are inac-

cessible in indoor applications, do not work in harsh environments, and are not sufficiently robust to jamming in military applications, [110]. Due to these and similar reasons, there has been great interest within the signal processing and robotic communities, among others, in developing efficient and low-cost localization algorithms that do not require GPS.

The solutions to localization, in general, consist of two phases: acquiring measurements and transforming them into coordinate information, [111]. In the first phase, nodes[1] collect measurements and exchange information with other network entities, including neighboring agents, anchors, or a combination of these. In the second phase, the information and measurements acquired are aggregated and used as inputs to a localization algorithm, [22]. The nature of the localization solution, i.e., whether *centralized* or *distributed*, depends on what types of measurements (such as estimates of relative distance between neighboring nodes or distance to a possibly distant anchors) are acquired and how the measurements and exchanged information are processed, for instance, whether by a central entity or in a distributed fashion using local computing at the nodes.

Several localization techniques have been proposed in the literature including successive refinements, [112, 113], maximum likelihood estimation, [114–

[1]The terminology node is being used in a generic sense here to denote network entities such as users, agents, anchors. The precise meaning and context will be clear from the mathematical formulations in the following.

116], multi-dimensional scaling, [117], optimization-based techniques, [118–120], probabilistic approaches, [121, 122], multilateration, [123, 124], and graph theoretical methods, [125, 126]. In general, the goal is to localize a network of nodes with unknown locations, in the presence of a number of anchors[2]. Most of these algorithms consider localization in static networks. However, due to the rapid advances in mobile computing and wireless technologies, mobility is becoming an important area of research in 5G IoT. Agents can be mobile or mounted on moving entities such as vehicles, agents, or humans. Mobility of nodes increases the capabilities of the network and creates the opportunity to improve their localization. It has been shown that the integration of mobile entities improves coverage, connectivity, and utility of the network deployment, and provides more data since more measurements can be taken while the agents move, [127, 128].

In fact, mobility plays a key role in the execution of certain applications such as target tracking, [129], traffic surveillance, [11], and environment monitoring, [12]. With advances in wearable technologies, there is a growing interest in localizing IoT objects particularly indoors where GPS-type locationing systems are not available, [130–132], which requires the development of self-localization algorithms that enable each object in the IoT to find its own

[2]Anchors are also referred to as reference nodes, seeds, landmarks, or beacons. In the remaining of this chapter, we call the nodes with known locations, anchors, and refer to all other nodes with unknown locations as agents.

location by implementing simple object-to-object communication over e.g., 5G, [16–18]. The mobility of these emergent mobile IoT devices makes the localization problem even more challenging as their locations and available neighborhoods (nearby nodes to implement peer-to-peer communication) keep changing. Therefore, it is important to design fast and accurate localization schemes for mobile networks without compromising the quality of embedded applications that require real-time location information.

Most localization algorithms for mobile networks use sequential Bayesian estimation or sequential Monte Carlo methods, see Appendix A for more details. In particular, the Bayesian methods use the recursive Bayes' rule to estimate the likelihood of an agent's location. Due to the non-linear nature of the involved conditional probabilities, the Bayesian solutions are generally intractable and cannot be determined analytically. Solutions involving extended Kalman filters [24–33], particle filters [34], and sequential Monte Carlo methods, [35–45], have been proposed to address the associated non-linearities and the intractable computation of the probability distributions. However, these approaches end up being sub-optimal and highly susceptible to the agents' initial guesses of their locations. Despite their implementation simplicity, Monte Carlo or particle filter methods are time-consuming as they need to continue sampling and filtering until enough samples are obtained to represent accurately the posterior distribution of an agent's position, [133], while requiring

a high density of anchors to achieve accurate location estimates.

As an alternative to the traditional non-linear approaches, in this chapter, we provide a distributed algorithm to solve localization in a network of mobile agents in the absence of any prior estimates of the initial locations. As a motivating scenario, consider a multi-agent network of ground/aerial vehicles with no central or local coordinator and with limited communications, whose task is to transport goods in an indoor facility, where GPS signals are not available. In order to perform a delivery task, each mobile agent has to know its own location first. In such settings, we are interested in developing distributed algorithms to track the agent locations such that the convergence is invariant to the initial position estimates. However, the implementation of distributed algorithms in mobile networks, where both agents and anchor(s) are moving, is not straightforward due to the following challenges: (i) no agent may be in proximity of any anchor; (ii) an agent may not be able to find nearby agents at all times to perform a distributed algorithm; and, (iii) the dynamic neighborhood at each agent results into a time-varying distributed algorithm, whose stability (convergence) analysis is non-trivial.

We address these challenges by providing an opportunistic update scenario, where an agent updates its location estimate in $\mathbb{R}^m$ only if it lies inside a convex hull of $m+1$ neighbors. We refer to such neighbors as a *triangulation set*. Using this approach, we show that the agent location estimates are improved as the

procedure continues and the algorithm tracks the true agent locations.

In this context, the main contribution of this chapter is to develop a *linear* framework for localization that enables us to circumvent the challenges posed by the predominant nonlinear approaches to this problem. This linear framework is not to be interpreted as a linearization of an existing nonlinear algorithm. Instead, the nonlinearity from range to location is embedded in an alternate representation provided by the barycentric coordinates.

We abstract our localization algorithm as an LTV system, whose system matrices may be stochastic or sub-stochastic. These system matrices do not belong to a finite or a countable set rendering many of the existing results inapplicable. In addition, since they can be either identity, stochastic or sub-stochastic, their product is no longer a group under multiplication. Thus, establishing the asymptotic behavior of the underlying LTV system is non-trivial. To address these challenges, we apply the results developed in Chapter 2 to study LTV convergence by partitioning the entire set of system matrices into slices and relating the convergence rate to the slice lengths. In particular, we show that the algorithm converges if the slice lengths do not grow faster than a certain exponential rate.

Next, since our localization scheme is based on the motion and the distance measurements, it is meaningful to evaluate the performance of the algorithm when these parameters are corrupted by noise. Hence, we study the impact

of noise on the convergence of the algorithm and provide modifications to counter the undesirable effects of noise. Finally, we investigate what role does motion play in the behavior of the localization algorithm and provide necessary conditions in terms of the number of nodes and the dimensions of motion required by our approach.

The rest of this chapter is organized as follows. In Section 4.2, we provide a comprehensive taxonomy of the existing localization algorithms and in particular discuss the emergence of mobility and the role it plays in networked localization. In Section 4.3, we formulate the localization problem in mobile networks. We propose our localization algorithm in Section 4.4, followed by the convergence analysis in Section 4.5. We investigate the effects of noise in Section 4.6, while in Section 4.7 we discuss the relationship between the dimension of motion and the minimum number of anchors required. We discuss different aspects of the algorithm in Section 4.9 and present detailed simulation results in Section 4.8. Finally, Section 4.10 concludes this chapter.

4.2 Taxonomy of localization approaches

In general, localization problems can be grouped into two main categories: position tracking and global localization. Position tracking, also known as local localization, requires the knowledge of the agents' starting locations, whereas

global localization refers to the process of determining agents' locations without any prior estimate of their initial locations. In the remaining of this chapter, we only consider the global localization problem. We now present a comprehensive taxonomy of existing localization algorithms.

Absolute vs. Relative: Absolute localization refers to the process of finding agent locations in a predetermined coordinate system, whereas relative localization refers to such process in a local environment, in which the nodes share a consistent coordinate system. In order to determine the absolute positions, it is necessary to use a number of anchors that have been deployed into the environment at known locations. These reference nodes define the coordinate system and contribute to the improvement of the estimated locations of the other agents in the network. The location estimates may be relative to a set of anchors at known locations in a local coordinate system, or absolute coordinates may be obtained if the positions of the anchors are known with respect to some global coordinate system, either via GPS or from a system administrator during startup, [134]. In relative localization, no such reference nodes exist and an arbitrary coordinate system can be chosen. Although increasing the number of anchors in general may lead to more accurate location estimates or may increase the speed of the localization process, the main issues with adding more anchors are that they make the process more expensive and often become useless after all the agents in the network have been localized.

Centralized vs. Distributed: In centralized localization, the locations of all agents are determined by a central coordinator, also referred to as *sink node*. This node first collects measurements (and possibly anchor locations), and then uses a localization algorithm to determine the locations of all the agents in the network and sends the estimated locations back to the corresponding agents. In contrast, such central coordinator does not exist in distributed localization, where each agent infers its location based on locally collected information. Despite higher accuracy in small-sized networks, centralized localization schemes suffer from scalability issues, and are not feasible in large-scale networks. In addition, comparing to the distributed algorithms, the centralized schemes are less reliable and require higher computational complexity due to, e.g., the accumulated inaccuracies caused by multi-hop communications over a wireless network. Distributed algorithms are preferred in applications where there is no powerful computational center to handle the necessary calculations, or when the large size of the network may lead to a communication bottleneck near the central processor, [110]. Therefore, there has been a growing effort in recent years to develop distributed algorithms with higher accuracies.

Conventional vs. Cooperative: In conventional approach, there is no agent-to-agent communication, and therefore agents do not play any role in the localization of other agents in the network. As an example, consider the

traditional *trilateration* for localization in $\mathbb{R}^2$. In this method, each agent first measures its distances to three anchors, and then finds its location as the intersection of three circles centered around each anchor. On the other hand, in cooperative approach agents take part in the localization process in a collaborative manner, and are able to exchange information with their neighboring nodes, which may include other agents with unknown locations as well as the anchors. Since anchors are the only reliable source of location information, the former method, also known as *non-cooperative* approach either requires the agents to lie within the communication radius of multiple anchors, which in turn requires a relatively large number of anchors, or to have long-range transmission capabilities in order to make measurements to the anchors. Cooperative localization removes such restrictions by allowing the inter-agent communications, which in turn increases the accuracy, robustness and the overall performance of the localization process.

Range-free vs. Range-based: The localization can also be divided into range-free and range-based methods, based on the measurements used for estimating agent locations. A large number of existing localization algorithms use distance and/or angle measurements to localize a network of agents, and are therefore referred to as range-based algorithms. Depending on the available hardware, the most common measurement techniques that are used in localization algorithms include Received Signal Strength (RSS), Time of Arrival

(ToA), Time Difference of Arrival (TDoA), and Angle of Arrival (AoA).

In particular, the distance estimates can be obtained from RSS, ToA, or TDoA measurements; RSS-based localization exploits the relation between power loss and the distance between sender and receiver, and does not require any specialized hardware. However, due to nonuniform signal propagation environments, RSS techniques suffer from low accuracy. More reliable distance measurements can be obtained by estimating the propagation time of the wireless signals, which forms the basis of ToA and TDoA measurements. ToA measures the difference between the time a signal is sent at the transmitter and the time it is received at the receiver. One-way (or round-trip) propagation time is then used to estimate the distances between neighboring nodes. ToA requires accurate synchronizion of the local time at the transmitter and at the receiver. TDoA technique is based on the idea that the location of an agent can be determined by examining the difference in time at which a specific signal arrives at multiple reference points. ToA and TDoA methods provide high estimation accuracy compared to RSS, but require additional hardware at the sensor nodes for a more complex process of timing and synchronization. Relative orientations can be determined using AoA measurements, which requires a node to be equipped with directional or multiple antennas. AoA measurements rely on a direct line-of-sight path from the transmitter to the receiver. In addition, in order to measure the traveled distance, acceleration,

and orientation, an agent may be equipped with an odometer or pedometer, an accelerometer, and an Inertial Measurement Unit (IMU), respectively, [22].

On the other hand, range-free algorithms, also known as proximity-based algorithms, use connectivity (topology) information to estimate the locations of the agents. The range-free localization schemes eliminate the need of specialized hardware on each agent, and are therefore less expensive. However, they suffer from lower accuracy compared to the range-based algorithms. Typical range-free localization algorithms include Centroid, [135], Amorphous, [123], DV-hop, [136], SeRLoc, [137], and APIT [138].

Static vs. Mobile: Localization literature has mainly focused on static networks where the nodes do not move and the network configuration does not change in time. This is particularly the case for WSNs, where the problem of locating mobile sensors has not been sufficiently addressed. However, due to the rapid advances in mobile computing and wireless communication technologies in recent years, mobility is becoming an important area of research in WSNs. In a particular class of WSNs, namely Mobile Wireless Sensor Networks (MWSNs), sensors can be mobile or mounted on moving entities such as vehicles, agents, humans, etc. In MWSNs, mobility of sensor nodes increases the capabilities of the network, and creates the opportunity to improve sensor localization. It has been shown, [127, 128], that the integration of mobile entities into WSNs improves coverage, connectivity, and utility of the sensor

network deployment and provides more data since more measurements can be taken while the sensors are moving. In addition, mobility plays a key role in the execution of an application. For example, in a MWSN that monitors wildfires, the mobile sensors can track the fire as it spreads, and stay out of its way, [134].

Mobility also enables sensor nodes to target and track the moving objects. Sensor tracking problem is an important aspect of many applications, including animal tracking, for the purpose of biological research, and logistics, e.g., to report the location of equipments in a warehouse when they are lost and need to be found, [110]. Another potential application of localization in mobile networks is the *Internet of Things* (IoT), which can be thought of as a massive network of objects such as sensors, agents, humans and electronic devices that are connected together and are able to collect and exchange data. With the advancements in wearable technologies, there is a growing interest in localizing IoT objects, [130–132], which requires the development of self-localization algorithms that enable each object in the IoT to find its own location by implementing simple object-to-object communications.

Despite all the aforementioned advantages, mobility can make the localization process more challenging; in statically deployed networks, the location of each agent needs to be determined once during initialization, whereas in mobile networks the agents must continuously obtain their locations as they

move inside the region of interest. Moreover, mobile nodes require additional power for mobility and are often equipped with a much larger energy reserve, or have self-charging capability that enables them to plug into the power grid to recharge their batteries, [134]. It is therefore crucial to design fast and accurate localization schemes for mobile networks without compromising the quality of applications that require wireless communications.

In the remaining of this chapter, our goal is to develop a cooperative, distributed, range-based algorithm to solve global localization in mobile networks.

4.3 Preliminaries and problem formulation

Consider a network of N mobile agents with unknown locations in the index set, Ω, and M possibly mobile anchors with known locations in the index set, κ. Let $\Psi = \Omega \cup \kappa$ be the set of all nodes (agents and anchors) located in $\mathbb{R}^m$, $m \geq 1$. We assume that the agents occupy a *non-trivial configuration*, i.e., all of them do not remain on a low-dimensional subspace in $\mathbb{R}^m$ effectively reducing the localization problem to $\mathbb{R}^{m-1}$. We define $\mathcal{N}_k^i \subseteq \Psi$ as the set of neighbors of agent, $i \in \Omega$, at time k. We denote the *true location* of the i-th node, $i \in \Psi$, at time k, with an m-dimensional row vector, $\mathbf{x}_k^{i*} \in \mathbb{R}^m$, where $k \geq 0$ is the discrete time index. The problem is to find the locations of the mobile agents in the set Ω, given any initialization of the underlying

algorithm.

We define the true distance between any two nodes, i and j, at the time of communication, k, as $\widetilde{d}_k^{ij}$. We assume that the measured distance, $\widehat{d}_k^{ij}$, at agent i includes noise, i.e.,

$$\widehat{d}_k^{ij} = \widetilde{d}_k^{ij} + r_k^{ij}, \tag{4.1}$$

where r_k^{ij} is the noise in the distance measurement at time k.

Motion model

There exist two main categories of motion models in the literature; *entity mobility models*, [139, 140], and *group mobility models*, [139, 141, 142]. The former describes the movement of each agent in the network independently from the other nodes, whereas the latter expresses the motion of a group of agents, possibly coordinated together by a leader or a central coordinator. Inspired by the unpredictable motion of many entities in nature, the Random Walk mobility model, [139], is a widely used entity mobility model. In this model, at each time step, each node moves from its current location to a new location by choosing a new direction and speed, randomly chosen from pre-defined ranges. A slightly different entity motion model is the Random Waypoint mobility model, [140], which includes a pause time at each iteration before the changes in speed and/or direction occur.

On the other hand, one of the earliest group mobility models is the Exponential Correlated Random model, [139]. In this mobility model, a motion function is created in order to generate the movements of the mobile nodes. Another example is the Column motion model, [141], which is mainly used in military applications. In this model, the nodes move in a straight line and predefined fixed directions. Reference Point group mobility model is also a common group mobility model that was introduced in [142]. According to this model, each group has a *logical center*. The center's motion defines the motion behavior of the entire group, including their location, direction, speed, acceleration, etc. Thus, the group trajectory is determined by providing a path for the center. The reader can refer to [143] for a survey and taxonomy on mobility models for vehicular ad hoc networks.

We consider Random Waypoint mobility model and express the motion as the deviation from the current to the next locations, i.e.,

$$\mathbf{x}_{k+1}^{i*} = \mathbf{x}_{k}^{i*} + \widetilde{\mathbf{x}}_{k+1}^{i}, \qquad i \in \Psi, \tag{4.2}$$

in which $\widetilde{\mathbf{x}}_{k}^{i}$ is the true motion vector at time k. We note here that the agents are assumed to move in a bounded region in $\mathbb{R}^m$ and hence $\widetilde{\mathbf{x}}_{k}^{i}$ cannot take values that take an agent outside this region. We assume that agent i measures a noisy version, $\widehat{\mathbf{x}}_{k}^{i}$, of this motion, e.g., by integrating the measurements

gathered by an accelerometer:

$$\widehat{\mathbf{x}}_k^i = \widetilde{\mathbf{x}}_k^i + n_k^i, \tag{4.3}$$

where n_k^i represents the integration noise at time k.

Barycentric representation

As we will explain in Section 4.4, to perform an update, each agent, say i, must find $m+1$ neighbors, such that it lies strictly in the interior of their convex hull, $\mathcal{C}(\cdot)$. An inclusion test to verify if an agent lies inside an available convex hull in a given time is described in Appendix B.1. We refer to a set of neighbors (of agent i) that pass the inclusion test at time k as a *triangulation set*, Θ_k^i. We note that the triangulation set, Θ_k^i, has a non-zero volume, i.e.,

$$A_{\Theta_k^i} > 0, \qquad \forall k \geq 0,$$

where $A_{\Theta_k^i}$ represents the m-dimensional volume, area in $\mathbb{R}^2$ or volume in $\mathbb{R}^3$, of $\mathcal{C}(\Theta_k^i)$, and can be computed by using the Cayley-Menger (CM) determinant, [144], see Appendix B.2. In $\mathbb{R}^m$, we have

$$i \in \mathcal{C}(\Theta_k^i) \;\text{ and }\; |\Theta_k^i| = m+1 \qquad \forall k \geq 0.$$

With the help of triangulation sets, we represent the location of agent i at time k, $\mathbf{x}_k^{i*}$, as a linear-convex combination of the locations of the neighboring

nodes in its triangulation set:

$$\mathbf{x}_k^{i*} = \sum_{j \in \Theta_k^i} a_k^{ij} \mathbf{x}_k^{j*}, \tag{4.4}$$

where a_k^{ij}'s are the *barycentric coordinates*, defined as

$$a_k^{ij} = \frac{A_{\Theta_k^i \cup \{i\} \setminus j}}{A_{\Theta_k^i}}. \tag{4.5}$$

The above representation dates back to the early work by Lagrange and Möbius, [145]. Since we assume agent i to lie strictly inside its triangulation set, all barycentric coordinates are strictly positive, i.e.,

$$a_k^{ij} > 0, \qquad \forall j \in \Theta_k^i.$$

Straightforward computation of CM determinants in $\mathbb{R}^2$ and $\mathbb{R}^3$, allows us to easily compute and take advantage of the barycentric coordinates in order to avoid nonlinearity in the localization process. Furthermore, note that the inclusion test only requires pairwise distances among the nodes in $\{i\} \cup \mathcal{C}_i(\cdot)$, see Section 4.9 for the associated computation complexity. Barycentric representation is illustrated in Fig. 4.1, where node i lies inside the convex hull of its neighbors at time k, i.e., nodes j, m and n. In this case, we can describe the true location of node i as follows:

$$\mathbf{x}_k^{i*} = a_k^{ij} \mathbf{x}_k^{j*} + a_k^{im} \mathbf{x}_k^{m*} + a_k^{in} \mathbf{x}_k^{n*}.$$

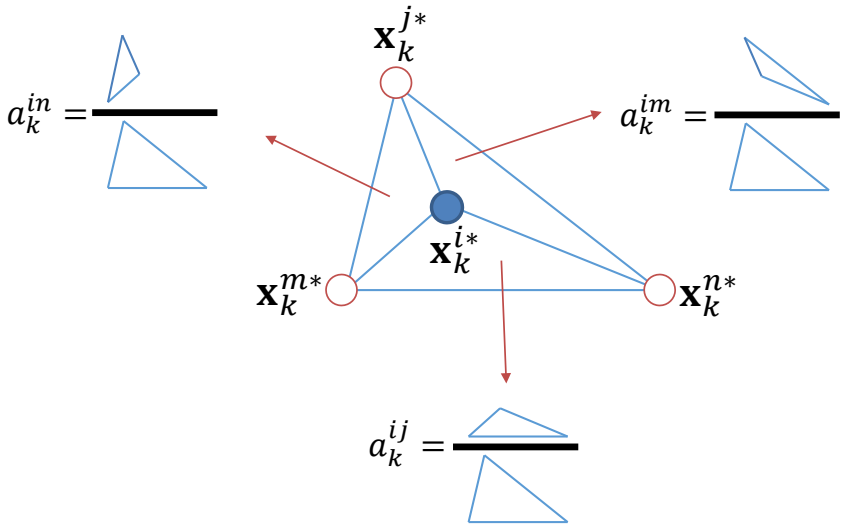

Figure 4.1: Barycentric representation.

Assumptions

We now enlist a set of assumptions, which we use later in the design and analysis of our localization algorithm:

B0: Anchor locations $\mathbf{u}_k^{m*}, m \in \kappa$, are known at all times.

B1: Each agent, $i \in \Omega$, measures a noisy distance, $\widehat{d}_k^{ij}$, to every node, $j \in \Psi$, in its communication radius, r, at time k, see Eq. (4.1).

B2: Each agent, $i \in \Omega$, knows a noisy version, $\widehat{\mathbf{x}}_k^i$, of its motion, $\widetilde{\mathbf{x}}_k^i$, at all times, see Eq. (4.3).

We assume that each agent is able to estimate its distances to the nearby nodes (agents and/or anchors) by using RSSI, ToA, TDoA, or camera-based

methods, [146, 147]. Under the above assumptions, we are interested in finding the true locations of each agent *without the presence of any central or local coordinator.*

In the following, Sections 4.4 and 4.5, we consider the ideal scenario when the motion and distance measurements are not effected by noise, i.e., $n_k^i = 0$ and $r_k^{ij} = 0$, in Eqs. (4.3) and (4.1). We then study the effects of noise in Section 4.6 and provide modifications to the algorithm to address the noise on the motion as well as the distance measurements.

4.4 Localization algorithm

Traditionally, localization has been treated as a non-linear problem that requires either: (i) solving circle equations when the agent-to-anchor distances are given; or, (ii) using law of sines to find the agent-to-anchor distances (and then solving circle equations) when the agent-to-anchor angles are given, [148]. The former approach is called *trilateration*, whereas the latter method is referred to as *triangulation*. The literature on localization is largely based on traditional trilateration and triangulation principles, or, in some cases, a combination of both, see [148] for a historic account; examples include [149–153].

In trilateration, the main idea is to first estimate the distances between an agent with unknown location and three anchors (in $\mathbb{R}^2$) and then to find

the location of the agent by solving three (non-linear) circle equations. This procedure is depicted in Fig. 4.2 for a static network in $\mathbb{R}^2$. In this figure, the unknown location is at the intersection of three circles, each circle is centered around an anchor, indicated by a red triangle, and the radius of each circle is the distance between the agent, represented by a solid circle, and the corresponding anchor. If the agent with unknown location can measure its dis-

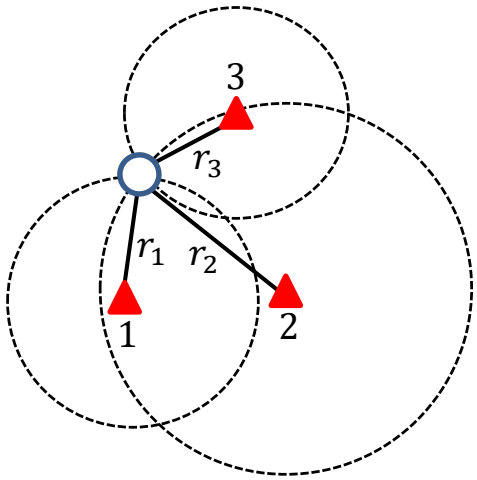

Figure 4.2: Traditional trilateration in $\mathbb{R}^2$.

tances, r_1, r_2, and r_3, to the three nodes with known locations, it can then find its location, $(x, y) \in \mathbb{R}^2$, by solving the following nonlinear circle equations:

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= r_1^2, \\ (x - x_2)^2 + (y - y_2)^2 &= r_2^2, \\ (x - x_3)^2 + (y - y_3)^2 &= r_3^2, \end{aligned} \tag{4.6}$$

where (x_1, y_1), (x_2, y_2), (x_3, y_3) represent the coordinates of the three anchors.

The placement of the anchors is arbitrary.

In a large network, like envisioned with IoT, trilateration would need each agent to find its distance to each of three anchors (in two dimensional space) and then solve the nonlinear equations (4.6). Clearly, for networks with large number of agents or in dynamic networks with mobile agents, this will unduly tax the system resources and be infeasible as it will require *either* placing a large number of anchors so that each agent finds at least three of them *or* long-distance communication and distance/angle estimates from the agents to a small number of possibly far-away anchors. To avoid these difficulties, in this section, we provide a distributed solution to localize a large number of agents in a mobile network. This localization framework can be thought of as a linear iterative solution to the nonlinear problem posed in terms of the circle equations in (4.6), which is applicable to networks with moving agents.

We now describe our localization algorithm[3]: At the beginning, each agent starts with a random guess of its location estimate (Alg. 1 line 2). At each time $k > 0$, we consider the following update scenarios for any arbitrary agent, $i \in \Omega$:

Case (i): If agent i does not find at least $m + 1$ neighbors, i.e., $0 \leq |\mathcal{N}_k^i| < m + 1$, it does not update its current location estimate (Alg. 1 lines 7 and 8).

[3]Main steps of the proposed localization algorithm are summarized in Algorithm 1

Case (ii): If agent i finds at least $m+1$ neighbors, i.e., $|\mathcal{N}_k^i| \geq m+1$, it performs the inclusion test, described in Appendix B.1, on all possible combinations of $m+1$ neighbors (Alg. 1 line 10). If agent i then fails to find a triangulation set, it does not update (Alg. 1 line 12), otherwise it applies the following update[4] (Alg. 1 line 14):

$$\mathbf{x}_{k+1}^i = \alpha_k \mathbf{x}_k^i + (1-\alpha_k) \sum_{j \in \Theta_k^i} a_k^{ij} \mathbf{x}_k^j + \widetilde{\mathbf{x}}_{k+1}^i, \tag{4.7}$$

where $\mathbf{x}_k^i$ is the location estimate of agent i at time k, Θ_k^i is the triangulation set at time k, $a_k^{ij} > 0$ is the barycentric coordinates of node i with respect to the node $j \in \Theta_k^i$, and α_k is such that

$$\alpha_k = \begin{cases} 1, & \forall k \mid \Theta_k^i = \emptyset, \\ \in [\beta, 1), & \forall k \mid \Theta_k^i \neq \emptyset, \end{cases} \tag{4.8}$$

in which β is a design parameter.

A network of $|\Omega| = 5$ mobile agents and $|\kappa| = 2$ anchors with fixed locations in $\mathbb{R}^2$ is illustrated in Fig. 4.3. In this figure, red triangles represent anchors and each circle represents an agent. To clarify the updating scenarios let us just consider agent 1, distinguished by a filled circle in Fig. 4.3, whose communication radius is represented by the red circles. At time k_j, Fig. 4.3 (a), agent 1 has only one neighbor and does not update, as per Case

[4]We consider two update scenarios for Case (ii); an agent can update with respect to all possible triangulation sets at time k, or alternatively, it can only update once as soon as the inclusion test is passed for the first time at time k.

Algorithm 1 Localize N agents in $\mathbb{R}^m$ in the presence of M anchors with precision p

$k \leftarrow 0$
$\mathbf{x}_0 \leftarrow$ Random initial coordinates
$\mathbf{e}_0 \leftarrow \mathbf{x}_0^* - \mathbf{x}_0$
while $\|\mathbf{e}_k\|_2 \geq 10^{-p}$ **do**
  $k \leftarrow k+$
  for $i = 1$ to N **do**
    if $0 \leq |\mathcal{N}_k^i| < m+1$ **then**
      Do not update
    else
      Perform the inclusion test on (all possible combinations of) $m+1$ neighbors
      if No triangulation set found **then**
        Do not update
      else
        Update location estimate according to Eq. (4.7)
      end if
    end if
  end for
  $\mathbf{e}_k \leftarrow \mathbf{x}_k^* - \mathbf{x}_k$
end while

(i). Although at time k_l, Fig. 4.3 (b), agent 1 finds $m+1 = 3$ nodes (two agents and one anchor) within its communication radius, it fails to update because the neighbors do not satisfy the inclusion test. In this case, $|\mathcal{N}_{k_l}^1| = 3$ and $\Theta_{k_l}^1 = \emptyset$. However, agent 1 updates its location estimates at time k_m and k_n, where in the former it finds two different triangulation sets: $\{2,3,4\}$ and $\{2,4,5\}$, and disregards $\{3,4,5\}$ and $\{2,3,5\}$. At any given time, k, note the difference between $\mathcal{N}_k^i = \emptyset$ and $\Theta_k^i = \emptyset$; while the former implies that node i has no neighbors, as in Case (i), the latter implies that agent i can-

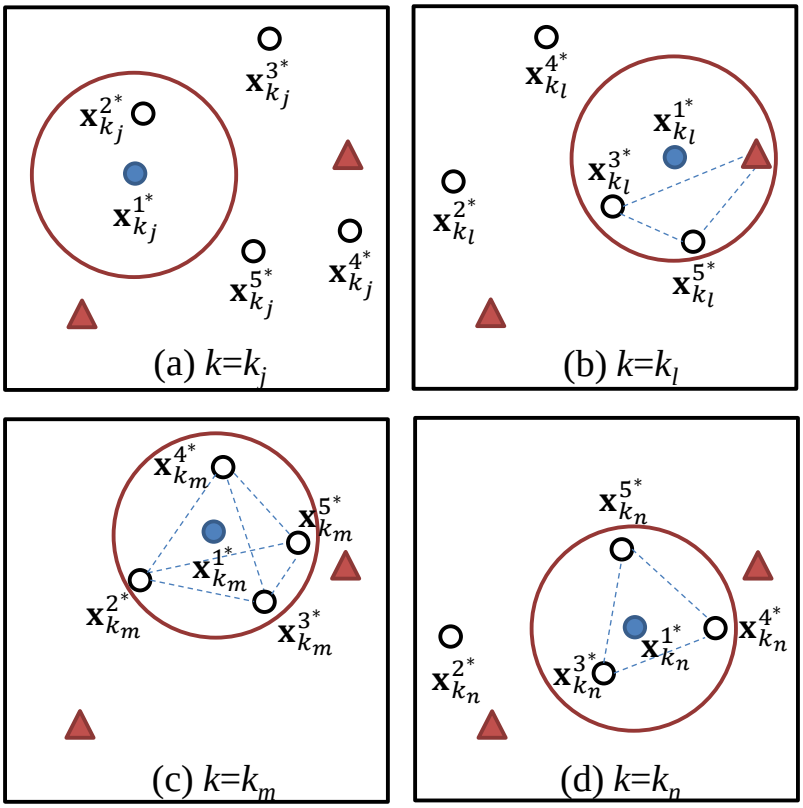

Figure 4.3: Update scenarios for a network of size 7.

not find any triangulation set among the neighbors.[5] For example at time k_j, Fig. 4.3 (a), we have $\mathcal{N}_k^i = \{2\}$ and $\Theta_k^i = \emptyset$, while at time k_n, Fig. 4.3 (d), we have $\mathcal{N}_k^i = \Theta_k^i = \{3, 4, 5\}$.

By separating the barycentric coordinates over the agents in Ω and the anchors in κ, we can split Eq. (4.7) as

$$\mathbf{x}_{k+1}^i = \alpha_k \mathbf{x}_k^i + (1 - \alpha_k) \left(\sum_{j \in \Theta_k^i \cap \Omega} p_k^{ij} \mathbf{x}_k^j \right),$$

[5]Note that our algorithm can be generalized to higher dimensions, e.g., 3-D, where triangles and areas become tetrahedrons and volumes, respectively.

$$+ \quad (1-\alpha_k)\left(\sum_{m\in\Theta_k^i\cap\kappa} b_k^{im}\mathbf{u}_k^{m*}\right)+\widetilde{\mathbf{x}}_{k+1}^i, \tag{4.9}$$

where $\mathbf{u}_k^{m*}$ is the true location of m-th anchor at time k and

$$a_k^{ij}=\begin{cases} p_k^{ij}, & \text{if } j\in\Theta_k^i\cap\Omega,\\ \\ b_k^{im}, & \text{if } m\in\Theta_k^i\cap\kappa. \end{cases} \tag{4.10}$$

Although it is possible that more than one agent finds a triangulation set and update at time k, in the remaining of this chapter we assume, without loss of generality, that at most one agent updates at each iteration. We can now write the above algorithm in matrix form as follows:

$$\mathbf{x}_{k+1}=\mathbf{P}_k\mathbf{x}_k+\mathbf{B}_k\mathbf{u}_k+\widetilde{\mathbf{x}}_{k+1}, \qquad k>0, \tag{4.11}$$

in which $\mathbf{x}_k\in\mathbb{R}^{N\times m}$ is the vector of agent coordinates at time k, $\mathbf{u}_k=\mathbf{u}_k^*\in\mathbb{R}^{M\times m}$ is the vector of anchor coordinates at time k, and $\widetilde{\mathbf{x}}_{k+1}\in\mathbb{R}^{N\times m}$ is the change in the location of agents at the beginning of the k-th iteration according to Eq. (4.2). Note that $\mathbf{P}_k$, and $\mathbf{B}_k$, the system matrix and the input matrix of the above LTV system, contain (weighted) barycentric coordinates at time k, with respect to the agents with unknown locations, and anchors, respectively.

Guaranteed anchor contribution: Since true information is only injected into the network by the anchors, we must set a lower bound on the weights assigned to the anchor states. Otherwise, the anchors may be assigned a weight that goes to zero over time, i.e., the anchors eventually are

excluded from the network. To make sure that anchors remain in the network, we naturally make the following assumption.

B3: When an anchor is involved in an update, i.e., for any $(\mathbf{B}_k)_{i,m} \neq 0$, we impose that

$$0 < \alpha \leq (\mathbf{B}_k)_{i,m}, \qquad \forall k, i \in \Omega, m \in \Theta_k^i \cap \kappa, \tag{4.12}$$

where α is the minimum anchor contribution, see Section 4.5.

The above assumption implies that if there is an anchor in the triangulation set, a certain amount of information is always contributed by the anchor. In other words, the update occurs only if the agent lies in an *appropriate* location inside the convex hull. By Assumption **B3**, the weight assigned to the anchor should be at least α. This weight comes from the barycentric coefficient corresponding to the anchor, i.e.,

$$\begin{aligned} (\mathbf{B}_k)_{i,m} &= (1-\alpha_k) b_k^{im} \\ &= (1-\alpha_k)\left(\frac{A_{\Theta_k^i \cup \{i\} \setminus m}}{A_{\Theta_k^i}}\right), \qquad m \in \kappa, \end{aligned} \tag{4.13}$$

where i and m indices represent the updating agent and an anchor, respectively. To clarify Assumption **B3**, we consider Fig. 4.4 where the updating agent lies inside the convex hull of three nodes, including two other agents and one anchor; let $\alpha = 0.25$ and $\alpha_k = 0$ for the sake of argument. Fig. 4.4 (Left) shows a situation, where the agent updates, and the anchor provides

the exact minimum contribution, i.e., the area of the shaded triangle is one fourth of the area of the convex hull triangle. On the other hand, if the agent

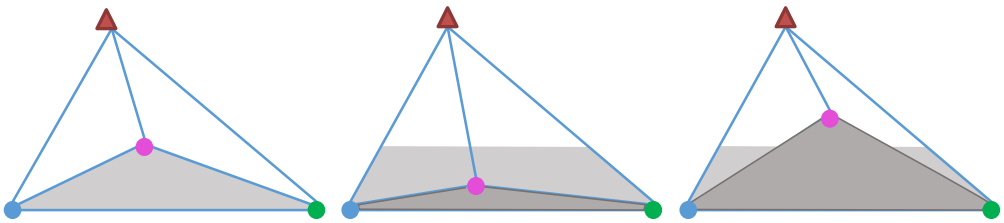

Figure 4.4: (Left) Marginal anchor contribution. (Middle) No update. (Right) Update with anchor.

lies within the shaded trapezoid illustrated in Fig. 4.4 (Middle), Eq. (4.13) becomes less than α, and Eq. (4.12) does not hold. No update occurs in this case since the anchor does not provide enough valuable information for the agent. The upper side of the shaded trapezoid is the threshold boundary, such that if the agent lies on this line, an update occurs and the anchor provides the exact minimum contribution. Thus, our measure to evaluate the contribution of an anchor in an update, is whether or not the updating agent lies inside a specific area, and not the distance between the agent and the anchor. If the updating agent stays inside the same convex hull, and moves closer to the anchor, as shown in Fig. 4.4 (Right), the agent performs an update and the weight assigned to the anchor exceeds α. Note that in this example we set $\alpha = 0.25$ only for the ease of demonstration. The localization algorithm works for any arbitrary small (non-zero) value of α, which corresponds to higher possibilities

of a successful update when an anchor is involved.

Minimum agent contribution: When there is no anchor in the triangulation set, we also restrict an agent from performing an update, unless it is located in a proper position inside a candidate convex hull. To this aim, we make the following assumption:

> **B4:** When there is no anchor in the triangulation set, for any agent, $j \in \Omega$, involved in an update, i.e., for any $(\mathbf{P}_k)_{i,j} \neq 0$, we assume that
>
> $$0 < \alpha' \leq (\mathbf{P}_k)_{i,j}, \qquad \forall k, j \in \Theta_k^i \cap \Omega, \tag{4.14}$$
>
> where i is the updating agent's index, and α' is the minimum agent contribution.

This assumption can be justified as we later consider the effects of noise on the localization algorithm. If an agent is located close to the boundaries of a convex hull, noise on distance measurements may lead to incorrect inclusion test results, i.e., a set of neighbors may fail the convexity test while the agent is indeed located inside the convex hull, or vice versa. To avoid such scenarios, we generalize Assumption **B3** to non-anchor agents in Assumption **B4**.

With the lower bounds on the self-weights according to Eq. (4.8), and the lower bounds on the weights assigned to the anchors and agents according to Eqs. (4.12) and (4.14), the matrix of barycentric coordinates with respect to agents with unknown locations, i.e., the system matrix at time k, $\mathbf{P}_k$, is either

(i) ***identity***, when no agent updates at time[6] k; or,

(ii) ***identity except a stochastic i-th row***, when an agent, say i, updates but finds no anchor in its triangulation set at time k, i.e.,

$$\Theta_k^i \neq \emptyset \text{ and } \Theta_k^i \cap \kappa = \emptyset; \text{ or,}$$

(iii) ***identity except a sub-stochastic i-th row***, when the updating agent, say i, has at least one anchor in its triangulation set at time k, i.e.,

$$\Theta_k^i \cap \kappa \neq \emptyset.$$

Note that when an agent, say i, performs an update at time k only with agents in the set Ω (Case (ii) above), the i-th row of the system matrix, $\mathbf{P}_k$, contains all of the non-zero weights (corresponding to the barycentric coordinates) and $\mathbf{B}_k$ is a zero matrix. Each row of $\mathbf{P}_k$ is identity except the i-th row whose sum is one due to convexity. However, if an anchor in the set κ is involved in an update (Case (iii) above), the i-th row of the input matrix, $\mathbf{B}_k$, contains a non-zero weight assigned to that anchor. Therefore the weights assigned to the other neighboring agents sum to a value strictly less than one; in other words, $\mathbf{P}_k$ is identity except the i-th row whose sum is strictly less than one making $\mathbf{P}_k$ sub-stochastic.

[6]Note that this case also characterizes the case, where an agent fails to update due to a (temporary) communication loss/drop.

In the next section, we use the results from Chapter 2 to provide sufficient conditions for the iterative localization algorithm, Eq. (4.11), to converge to the true agent locations.

4.5 Convergence analysis

We start this section by reviewing the main results on the convergence of (sub-) stochastic LTV systems that was provided in Chapter 2. We will then adapt these results to investigate the convergence of our localization algorithm represented by Eq. (4.11). Recall the following Assumptions from Section 2.4 on the update at time k:

A0: When the updating agent, i, has no neighboring anchor, we have

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} = 1, \tag{4.15}$$

resulting in a stochastic $\mathbf{P}_k$.

A1: When the updating agent, i, has no anchor but at least one agent as a neighbor, the weights are such that

$$0 < \beta_1 \leq (\mathbf{P}_k)_{i,l} < 1, \qquad \forall l \in \mathcal{D}_k^i,\ \beta_1 \in \mathbb{R}. \tag{4.16}$$

A2: When the updating agent updates with an anchor, we have

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} \leq \beta_2 < 1 \tag{4.17}$$

resulting in a sub-stochastic $\mathbf{P}_k$.

By defining a slice, $\mathbf{M}_j$, as the smallest product of consecutive system matrices, such that: (i) the infinity norm of each slice is less than one; *and*, (ii) the entire sequence of system matrices is covered by non-overlapping slices, the following theorem from Section 2.4 characterizes the asymptotic behavior of (sub-) stochastic LTV systems under the above Assumptions:

Theorem 1. *With Assumptions **A0-A2**, the following LTV system*

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k, \tag{4.18}$$

converges to zero if any one of the following holds:

(i) *Each slices has a bounded length, i.e.*

$$|\mathbf{M}_j| \leq N < \infty, \forall j,\ N \in \mathbb{N}; \tag{4.19}$$

(ii) *There exists a set, $\boldsymbol{J}_1$, consisting of an infinite number of slices such that*

$$|\mathbf{M}_j| \leq N_1 < \infty, \forall \mathbf{M}_j \in \mathbf{J}_1, \text{ and } |\mathbf{M}_j| < \infty, \forall \mathbf{M}_j \notin \mathbf{J}_1; \tag{4.20}$$

(iii) *For every $i \in \mathbb{N}$, there exists a set, $\boldsymbol{J}_2$, of slices such that*

$$\exists \mathbf{M}_j \in \mathbf{J}_2 : \quad |\mathbf{M}_j| \leq \frac{1}{\ln(\beta_1)} \ln\left(\frac{1 - e^{(-\gamma_2 i^{-\gamma_1})}}{1 - \beta_2}\right) + 1, \tag{4.21}$$

for some $\gamma_1 \in [0, 1]$, $\gamma_2 > 0$, and $|\boldsymbol{M}_j| < \infty, j \notin J_2$.

In what follows, we use the results of the above theorem to study the convergence of Eq. (4.11). We continue this section with the following lemma:

Lemma 8. *Under Assumptions **B0-B4** and no noise, the product of system matrices, $\mathbf{P}_k$'s, in the LTV system represented by Eq.* (4.11)*, converges to zero if any one of the three conditions in Theorem 1 holds.*

Proof. We need to show that Assumptions **A0-A2** holds for the LTV system, Eq. (4.11), that captures the localization algorithm. First note that Assumption **A0** is followed from the fact that barycentric coordinates always sum to one. Also note that Eqs. (4.8) and (4.14) result in Assumption **A1** if we set $\beta_1 = \min\{\beta, \alpha'\}$. On the other hand, if Assumption **B3** holds, i.e., if there exist at least one anchor in the triangulation set, we can write

$$\sum_{j \in \Theta_k^i \cap \Omega} p_k^{ij} = 1 - \sum_{m \in \Theta_k^i \cap \kappa} b_k^{im}, \tag{4.22}$$

in which we again used the fact that barycentric coordinates sum to one. Also, from Eq. (4.9), we have

$$\alpha_k + \sum_{j \in \Theta_k^i \cap \Omega} (1-\alpha_k) p_k^{ij} = 1 - \sum_{m \in \Theta_k^i \cap \kappa} \underbrace{(1-\alpha_k) b_k^{im}}_{=(\mathbf{B}_k)_{i,m}}. \tag{4.23}$$

Note that the second term on the right hand side of Eq. (4.23) gives the sum of all weights assigned to the anchor(s) in the triangulation set. This term is minimized, and hence the right hand side of Eq. (4.23) is maximized, when there is only one anchor among the neighbors of the updating agent, and the

minimum weight, α, is assigned to this anchor according to Eq. (4.12). In this case, using Eqs. (4.9) and (4.23), we can write

$$\sum_{j\in\Theta_k^i\cap\Omega} (\mathbf{P}_k)_{i,j} \leq 1-\alpha < 1, \tag{4.24}$$

which provides an upper bound on the i-th row sum of $\mathbf{P}_k$, and results in Assumption **A2** if we choose $\beta_2 = 1-\alpha$. With Assumptions **A0-A2** satisfied, the rest of the proof follows from that of Theorem 1. □

We now provide our main result in the following theorem.

Theorem 3. *Consider a network of M (possibly mobile) anchors and N mobile agents moving in a finite and bounded region. Then, under Assumptions **B0-B4** and no noise, for any (random or deterministic) motion that satisfies one of the conditions in Theorem 1, the solution of Eq.* (4.11) *asymptotically converges to the true agent locations.*

Proof. The motion of the agents in a finite, bounded region results in the following LTV system:

$$\mathbf{x}_{k+1} = \mathbf{P}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \widetilde{\mathbf{x}}_{k+1}, \qquad k>0, \tag{4.25}$$

in which the system matrices, $\mathbf{P}_k$'s switch between stochastic and sub-stochastic matrices. Therefore, under Assumptions **B0-B4** and according to Lemma 8, if any of the conditions in Theorem 1 is satisfied, we have

$$\lim_{k\to\infty}\prod_{l=0}^{k}\mathbf{P}_l = \mathbf{0}_{N\times N}. \tag{4.26}$$

On the other hand, the true agent locations are given by:

$$\mathbf{x}^*_{k+1} = \mathbf{P}_k \mathbf{x}^*_k + \mathbf{B}_k \mathbf{u}_k + \widetilde{\mathbf{x}}_{k+1}. \tag{4.27}$$

To justify the above equation, note that if agent i lies inside the convex hull of $m+1$ neighbors at time k, its true location can be expressed as a convex combination of the *true* locations of its neighbors. Note that when the inclusion test is not passed at any agent at time k, the system matrix, $\mathbf{P}_k$, and the input matrix, $\mathbf{B}_k$, become identity and zero matrices, respectively, and the location estimate, $\mathbf{x}_{k+1}$, and the true locations, $\mathbf{x}^*_{k+1}$ in Eqs. (4.25) and (4.27) update according to the motion vector. By subtracting Eq. (4.25) from Eq. (4.27) the error dynamics can be obtained as follows

$$\mathbf{e}_{k+1} \triangleq \mathbf{x}^*_{k+1} - \mathbf{x}_{k+1} = \mathbf{P}_k(\mathbf{x}^*_k - \mathbf{x}_k) = \mathbf{P}_k \mathbf{e}_k, \tag{4.28}$$

which converges to zero from Lemma 8 and Eq. (4.26), and the proof is complete. □

Discussion

In what follows, we shed some light on Theorem 1 in the context of our localization algorithm, and elaborate on the choice of design parameters.

(i) As mentioned earlier, each slice is initiated with a sub-stochastic update, which only occurs when an anchor is among the neighbors of the updating

agent. In other words, to initiate a slice an agent with unknown location must have an anchor in its triangulation set. Thus, the first system matrix of each slice has one sub-stochastic, and $N-1$ stochastic rows. We call an agent *informed* after it updates with an anchor. Therefore, at the beginning of a slice there are one informed and $N-1$ *uninformed* agents in the network.

(ii) A slice is terminated after all rows of a slice are sub-stochastic, i.e., all agents are informed. For this to happen, each agent has to either update with an anchor directly, or indirectly, i.e., update with an agent who updated with an anchor in the same slice. By *indirectly*, we mean that an agent receives information not from an anchor but by any other informed agent in the network. Completion of each slice corresponds to the propagation of location information from anchor(s) to all agents in the network, and the location estimates are refined each and every time a slice is completed. Hence, slice representation is closely related to the information flow from the anchors to every other agent in the network.

(iii) Theorem 1 provides the conditions on the rate, at which such information should propagate for convergence to the true agent locations. The first two cases in Theorem 1 require all or an infinite subset of slices to have bounded lengths. In other words, the information from the anchors has to reach all agents within L or L_1 iterations, infinitely often. These conditions are relaxed in the third case; Eq. (4.21) implies that all we need is an infinite

subset of slices whose lengths grow slower than a certain exponential rate. The condition on motion described in Theorem 3 can be translated to a condition on the information dissemination from Theorem 1. Thus, any motion that guarantees this information dissemination suffices.

(iv) The non-zero self-weights, α_k's, assigned to the previous state of the updating agent guarantees that an informed agent does not become uninformed again within the same slice, e.g., by performing an update with a set of uninformed agents before the slice is complete.

(v) Although the proposed algorithm converges for any value of $0 < \alpha < 1$, the convergence rate of the algorithm is affected by the choice of α: By choosing α arbitrarily close to 1, an agent has to get arbitrarily close to an anchor in order to perform an update with respect to that anchor (see Fig. 4.4), which in turn corresponds to arbitrary large number of iterations for the termination of each slice. On the other hand, setting α arbitrarily close to zero makes slice norms arbitrarily close to 1. A proper choice can be made by considering the motion model, the communication protocol, and the number of available anchors in the network.

4.6 Localization under imperfect measurements

The noise on the motion and distance measurements degrades the performance of the localization algorithm, as expected, and in certain cases the location error is larger than the region of motion; this is shown experimentally in Section 4.8. In what follows, we provide the modifications, **M1-M3**, to the proposed algorithm to counter the undesirable effects of noise in case of motion, $\widehat{\mathbf{x}}_k^i$, and on the distance measurements, $\widehat{d}_k^{ij}$.

Discard unreliable Cayley-Menger determinants

To get meaningful values for the areas and volumes in $\mathbb{R}^2$ and $\mathbb{R}^3$ according to Eq. (B.3), the corresponding Cayley-Menger determinants computed with perfect distance measurements must obey a certain sign in order for the square root to convey a meaningful result (negative in $\mathbb{R}^2$ and positive in $\mathbb{R}^3$). Therefore, we suggest the first modification to the algorithm as follows:

M1: An agent does not perform an inclusion test if the corresponding Cayley-Menger determinant is positive in $\mathbb{R}^2$, or negative in $\mathbb{R}^3$.

Inclusion test error

Even in the case of perfect distance measurements, the inclusion test results may not be accurate due to the noise on the motion, which in turn corresponds to imperfect location updates *at each and every iteration*. To address this issue, we propose the following modification to the algorithm:

M2: Suppose the inclusion test is passed at time k by a triangulation set, Θ_k^i. agent i performs an update only if

$$\epsilon_k^i = \left| \frac{\sum_{j \in \Theta_k^i} A_{\Theta_k^i \cup \{i\} \setminus j} - A_{\Theta_k^i}}{A_{\Theta_k^i}} \right| < \epsilon, \tag{4.29}$$

where ϵ_k^i is the *relative inclusion test error* at time k for agent i, and ϵ is a design parameter. The optimal choice of ϵ depends on the number of agents and anchors in the network and the statistics of the noise.

Convexity

Finally, in order to guarantee the convexity in the updates in the presence of noise, we consider the following modification:

M3: Suppose the inclusion test is passed by a triangulation set, $\Theta_k^i = \{j,\ l,\ m\}$, and Eq. (4.29) holds at time k. Agent i first computes, a_k^{ij}, a_k^{il}, and a_k^{im}, according to Eq. (4.5) and using the noisy distance measurement. It then normalizes the weights assigned to each neighbor in order

to preserve the convexity of the update. For example, the (normalized) weight assigned to agent j is computed as $a_k^{ij} / \sum_{n \in \Theta_k^i} a_k^{in}$.

In Section 4.8, we empirically show that the above modifications to the algorithm improve the results significantly in presence of noise, and result in a bounded error in location estimates.

4.7 How many anchors are necessary?

In this section, we investigate the minimal number of anchors required for localizing an arbitrary number of agents using the algorithm described in Section 4.4. We denote the subspace of motion at agent, $i \in \Omega$, and anchor, $j \in \kappa$, by $\mathcal{M}_i$ and $\mathcal{U}_j$, respectively. Let us clarify this notation with a simple example: Suppose agent 1 is moving along a vertical line (regardless of the direction); this line forms $\mathcal{M}_1$, and $\dim \mathcal{M}_1 = 1$. Note that $\mathcal{M}_i$ or $\mathcal{U}_j$ includes all possible locations that the i-th agent or the j-th anchor occupies throughout the localization process, i.e., discrete times $k = 1,\ 2, \ldots$. Now consider another agent, 2, which is moving along a vertical line parallel to $\mathcal{M}_1$ (regardless of the direction); in this case we will have

$$\dim \cup_{i=1,2} \mathcal{M}_i = \dim \mathcal{M}_2 = 1.$$

However, if the two lines are linearly independent, they span $\mathbb{R}^2$, and we will have

$$\dim \cup_{i=1,2} \mathcal{M}_i = 2.$$

Assuming $\mathbb{R}^m$, we show that the motion of the agents and anchors in $l \leq m$ dimensions allows us to reduce the number of anchors from $m+1$ by l. Note that the traditional trilateration scheme requires at least 3 nodes with known locations in $\mathbb{R}^2$. Therefore, assuming $m+1$ anchors in $\mathbb{R}^m$ has been standard in all range-based localization algorithms in the literature, see e.g., [151, 154]. In the following theorem, we develop necessary conditions for the proposed algorithm to track the true location of agents. Since we can provide agents with up to m degrees of freedom in their motion in $\mathbb{R}^m$, we then show that our localization algorithm works in the presence of only one $(m+1-m)$ anchor.

Theorem 4. *For the LTV dynamics in Eq.* (4.11) *to track the true agent locations, following a non-trivial configuration in* $\mathbb{R}^m, m = 2$*, the following conditions must be satisfied*

$$|\kappa| \geq 1, \tag{4.30}$$

$$|\kappa| + |\Omega| \geq m+2, \tag{4.31}$$

$$|\kappa| + \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j \geq m+1. \tag{4.32}$$

Proof. Eq. (4.30) is trivial, because when there is no anchor in the network, all system matrices become stochastic, and the error dynamics in Eq. (4.28)

do not converge to zero. This is equivalent to Case (ii) in Section 4.4, where the update is always in terms of agents, i.e., $\mathbf{B}_k$ is always zero in Eq. (4.11). Eq. (4.31) stems from the fact that each agent requires at least $m+1$ neighbors to perform an update in $\mathbb{R}^m$, assuming non-trivial configurations. In order to prove the necessary condition in Eq. (4.32), let us first consider the 2-dimensional Euclidean space. Suppose on the contrary that Eq. (4.32) does not hold. In this case, there exist four possible scenarios as follows:

(i) $|\kappa| = 0, \text{and} \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i = 2$; or,

(ii) $|\kappa| = 1, \text{and } \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j = 0$; or,

(iii) $|\kappa| = 1, \text{and } \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j = 1$; or,

(iv) $|\kappa| = 2, \text{and } \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j = 0$.

Localization is not possible in Case (i), as it violates Eq. (4.30). Case (ii)– with 1 anchor, 3 agents and no motion, is where at most one agent may be able to lie inside the convex hull of the remaining three; making the triangulation of the second agent impossible. Same argument can be applied for Case (iv)– with 2 anchors, 2 agents and no motion. Mathematically, these two cases mean that a slice is never completed because the rows of $\mathbf{P}_k$ in Eq. (4.11) corresponding to the non-updating agents will always be the corresponding rows of identity. Thus, all we need to show is that localization is not possible

in Case (iii), where there is one anchor in the network, and the dimension of the motion in all nodes is one, i.e., all nodes can only move along parallel lines in $\mathbb{R}^2$.

We start with the best possible scenario, which initially allows one agent to perform an update. We then choose an arbitrary direction in which all nodes

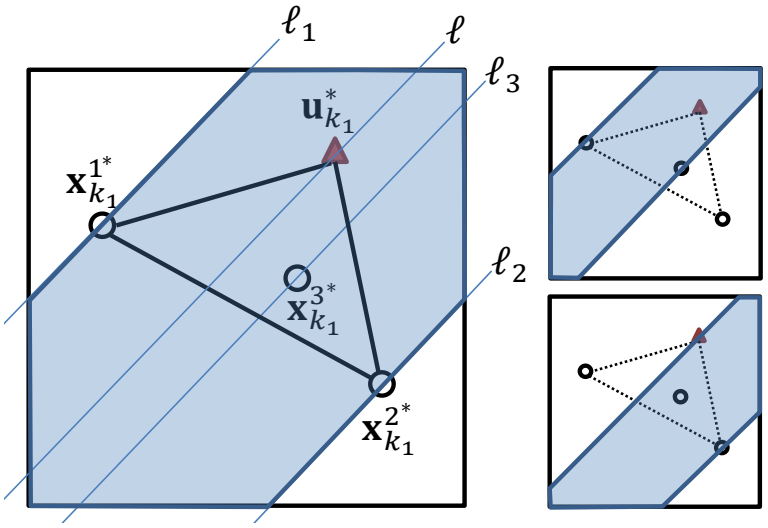

Figure 4.5: Only agent 1 can update in this configuration.

are allowed to move. As shown in Fig. 4.5 (Left), the anchor and the three agents can move along ℓ, ℓ_1, ℓ_2, and ℓ_3, respectively. If this direction is chosen such that the anchor's line of motion lies within the strip between ℓ_1 and ℓ_2 (the lines of motion for the two agents that are not initially able to update), no other agent will be able to perform an update. That is because agents 1 and 2 have to move inside the shaded regions illustrated in Fig. 4.5 (Right) in order to (possibly) perform an update. Clearly, any vector that provides such

motion is linearly independent of the agents' direction of motion.

We now consider the case where the direction of motion is such that the anchor lies outside the aforementioned strip. This scenario is illustrated in Fig. 4.6. At time k_2, agent 3 lies inside the convex hull of the anchor and the

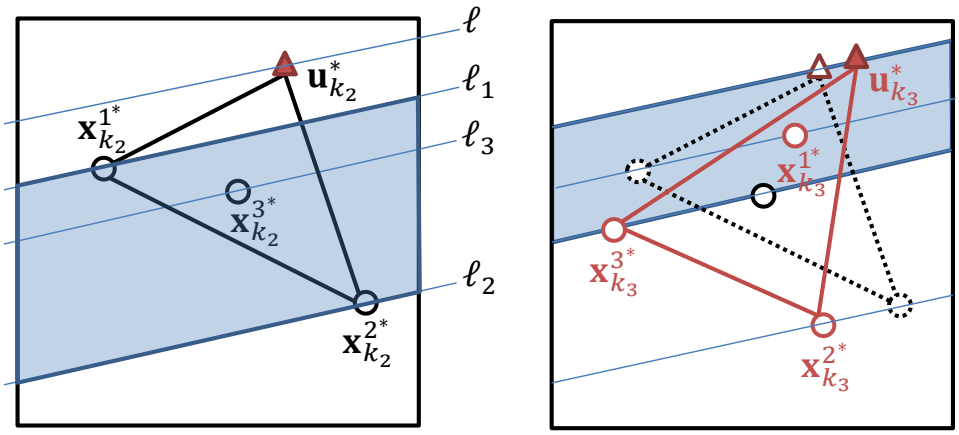

Figure 4.6: (Left) Agent 3 updates at time k_2. (Right) Agent 1 updates at time k_3. Agent 2 can not update in this configuration.

other two agents, and is able to perform an update, see Fig. 4.6 (Left). Due to the motion, Fig. 4.6 (Right), the agents move along the specified parallel lines such that at time k_3, agent 1 lies inside the triangle of the anchor and agents 2 and 3. However, the farthest agent from the anchor (agent 2 in this case) will never be able to perform an update, unless it moves inside the shaded region shown in Fig. 4.6 (Right), i.e., the strip between the lines of motion corresponding to agent 3 and the anchor. This in turn requires a motion vector that is linearly independent of the agents' specified direction of motion,

i.e., $\dim \underset{i\in\Omega}{\cup} \mathcal{M}_i$ greater than 1. Mathematically, row 2 of $\mathbf{P}_k$ in Eq. (4.11) never changes and a slice never completes. □

Although Theorem 4 is applicable to $\mathbb{R}^3$, a formal proof is beyond the scope of this thesis. In the sequel, we only give a sketch of the proof to show that localization is not possible with one anchor and two dimensional motion in $\mathbb{R}^3$. This scenario is shown in Fig. 4.7, where agent 4 initially lies inside the convex hull (a tetrahedron in $\mathbb{R}^3$) of the other nodes. The anchor and agents

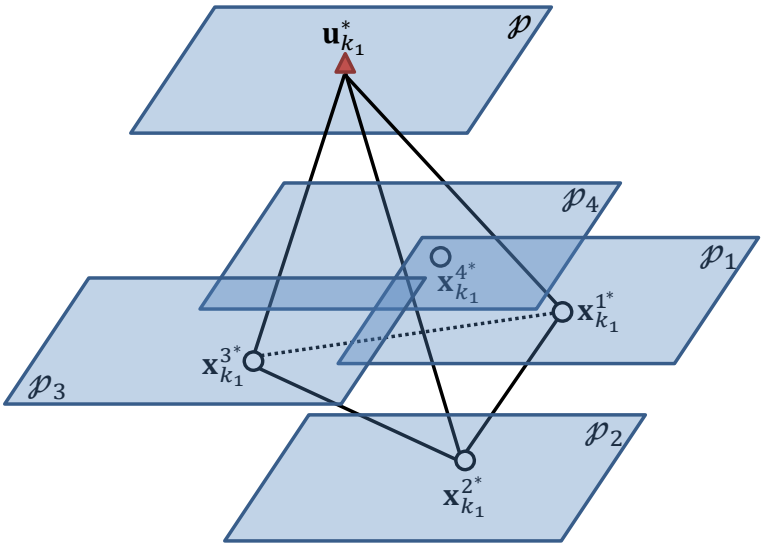

Figure 4.7: Localization with one anchor and 2-dimensional motion is not possible in $\mathbb{R}^3$.

$1-4$ can move on the planes $\mathcal{P}$, and $\mathcal{P}_1-\mathcal{P}_4$ as shown in Fig. 4.7. Clearly, the farthest agent from the anchor, agent 2 in this case, is not able to find a triangulation set, unless it moves into the space between $\mathcal{P}_3$ and $\mathcal{P}$, which in turn requires a motion vector which is not in the span of any pair of linearly

independent vectors in the motion planes, and thus makes localization of agent 2 impossible.

Theorem 4 provides necessary conditions for our localization algorithm in terms of the minimal number of anchors and the dimension of motion in the network. In what follows, we show how adding a new anchor or a new

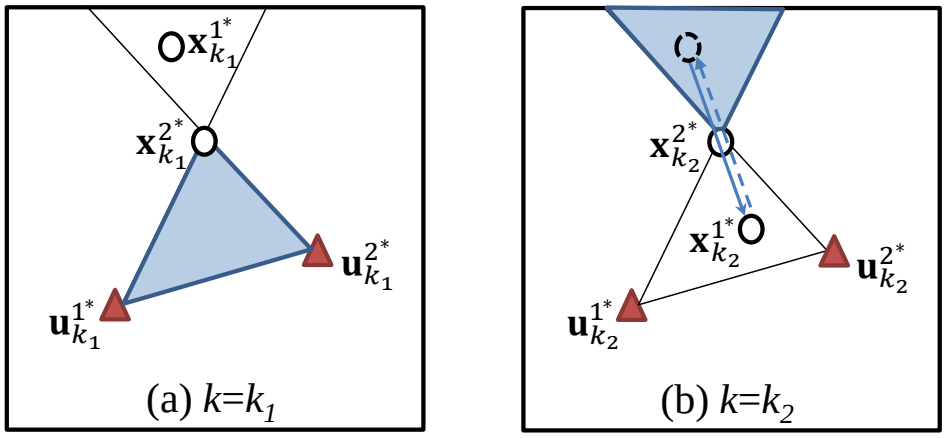

Figure 4.8: Localization with two anchors and one dimensional motion in $\mathbb{R}^2$.

dimension in the motion make the localization algorithm work in $\mathbb{R}^2$. We start with Case (ii) in Theorem 4 and add one more anchor, i.e.,

$$|\kappa| = 2, \text{ and } \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j = 1.$$

This scenario is illustrated in Fig. 4.8. Without loss of generality, we assume that only one agent, 1, is moving from time k_1 to k_2. To perform an update, this agent has to lie inside the convex hull of the two anchors and agent 2, the shaded region in Fig. 4.8 (a). To this aim, one possible motion vector is shown

in Fig. 4.8 (b) with a solid vector. Note that in order to let agent 2 update, agent 1 can move via the same motion vector (and in the opposite direction) to lie inside the shaded region shown in Fig. 4.8.

We now revisit Case (ii) in Theorem 4 and add one more motion dimension, i.e.,

$$|\kappa| = 1, \text{and } \dim \underset{i\in\Omega}{\cup} \mathcal{M}_i + \dim \underset{j\in\kappa}{\cup} \mathcal{U}_j = 2.$$

A possible motion trajectory of one agent, 1, which leads to the triangulation of all agents in a network of size 4 is shown in Fig. 4.9, assuming that the anchor and all the agents, except, agent 3, are static. We denote the motion vectors with $\mathbf{v}_1$ and $\mathbf{v}_2$, which are orthogonal in this case. The shaded regions in Fig. 4.9 (a), (b), and (c) represent the locations, where agents 2, 1, and 3 can update. For example, agent 2 lies inside a triangle of agents 3, agent 1, and the anchor, only if agent 3 moves to the shaded region in Fig. 4.9 (a) between time k_1 and k_2. Note that Fig. 4.9 illustrates a scenario, in which one slice is completed, i.e., each agent receives information from the anchor at least once. Clearly, to achieve the convergence to the true locations, such motion has to be repeated infinitely often such that one of the conditions in Theorem 1 is satisfied.

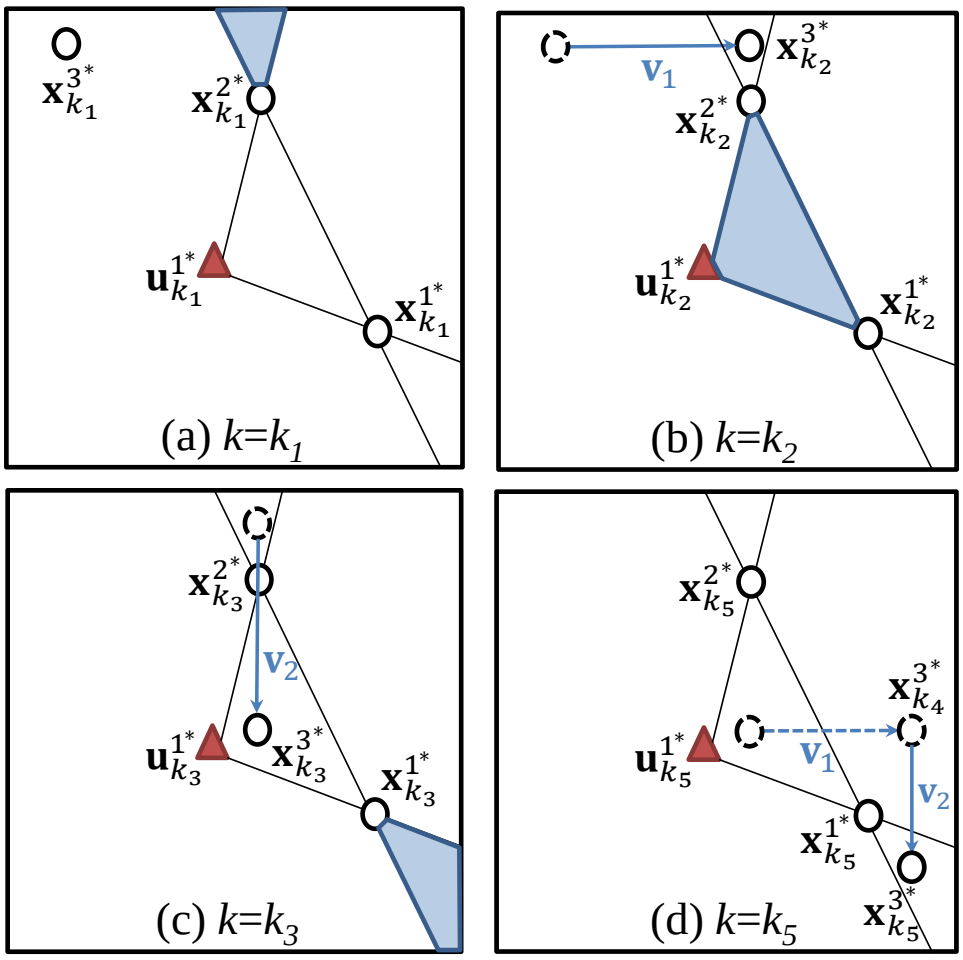

Figure 4.9: Localization with one anchor and two dimensional motion.

4.8 Simulations

In this section, we provide the simulation results to illustrate the proposed localization algorithm in $\mathbb{R}^2$. Let the true location of the i-th agent, $i \in \Omega$ in $\mathbb{R}^2$, be decomposed as $\mathbf{x}_k^{i*} = [x_k^{i*} \; y_k^{i*}]$. We consider the following random motion model for agent $i \in \Omega$:

$$x_{k+1}^{i*} = x_k^{i*} + d_{k+1}^i \cos(\theta_{k+1}^i), \quad y_{k+1}^{i*} = y_k^{i*} + d_{k+1}^i \sin(\theta_{k+1}^i), \tag{4.33}$$

in which d^i_{k+1} and θ^i_{k+1} denote the distance and angle traveled by agent i, between time k and $k+1$, and are random. We choose, d^i_k and θ^i_k, to have uniform distribution over the intervals of $[0\ d_{\max}]$ and $[0\ 2\pi]$, respectively, such that each agent, $i \in \Omega$, does not leave the bounded region of interest. Also note that the random motion is assumed to be statistically independent among the agents. In what follows, we first provide the simulation results in noiseless scenarios, and then study the effects of noise and the proposed modifications in the convergence of the algorithm.

Localization in noiseless scenarios

We first consider a network of 5 mobile agents with unknown locations, and only one anchor, whose location is fixed and perfectly known at all times. In

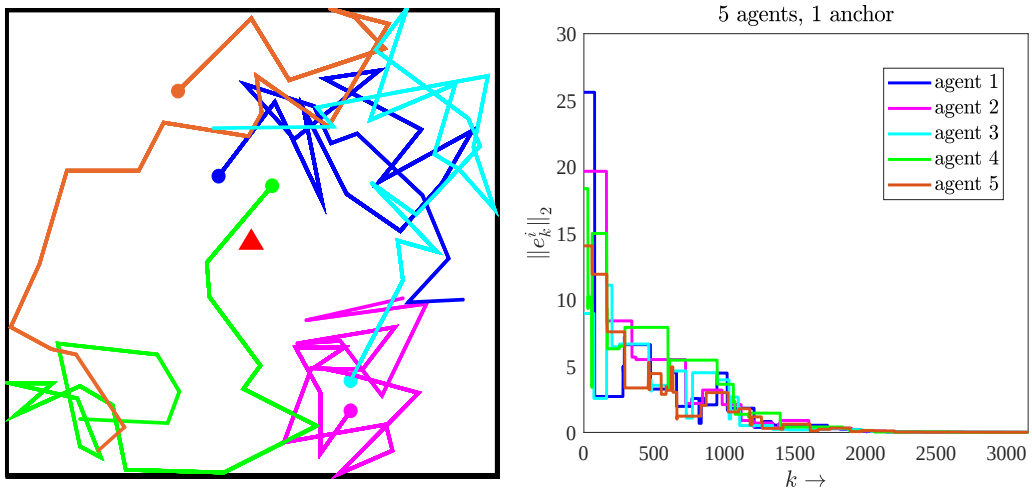

Figure 4.10: (Left) Motion model. (Right) Convergence.

the beginning, all agents are randomly deployed inside a 20m $\times$ 20m square. We fix the location of the anchor in the center of the region. For all simulations in this section, we set the communication radius to $r = 2$m and $d_{\max} = 5$m. All agents are initially assigned with a random estimate of their initial locations. If an agent finds at least $m+1 = 3$ neighbors, it performs the inclusion test, as described in Appendix B.1. If the opportunity occurs, i.e., if an agent finds a triangulation set among the neighboring nodes, it updates its location estimate according to Eq. (4.9). The agent does not perform any update otherwise. Throughout this section, we consider the scenario, where an agent performs multiple updates at each iteration with respect to all possible triangulation sets found in that iteration. To ensure that the updating agent retains the valuable information it may have received from the anchor, and to guarantee a minimum contribution by the anchor when it is involved in an update, we set $\alpha_k = \beta = 0.01$ and $\alpha = \alpha' = 0.01$, respectively. Fig. 4.10 (Left) shows the motion model that we choose as random according to Eq. (4.33), i.e., the trajectories of $N = 5$ mobile agents for the first 20 iterations. We choose the second norm of the error vector, $\mathbf{e}_k^i$, to characterize the convergence, i.e.,

$$\|\mathbf{e}_k^i\|_2 = \sqrt{(x_k^i - x_k^{i*})^2 + (y_k^i - y_k^{i*})^2}. \tag{4.34}$$

The algorithm converges to the true agent locations as $\|\mathbf{e}_k^i\|_2 \to 0, \forall\ i$. The convergence of the algorithm in this case is illustrated in Fig. 4.10 (Right) for

one simulation. In Fig. 4.11 (Left), we provide the convergence results for the networks with one anchor and 5, 10, 20 and 100 agents with unknown locations. Each curve indicates the average over $n = 20$ Monte Carlo simulations. Due to

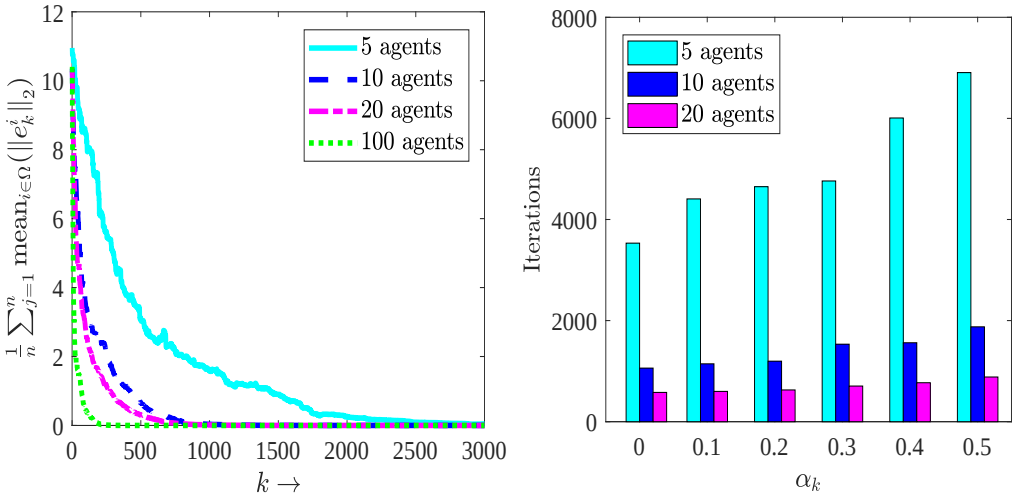

Figure 4.11: (Left) Convergence of networks with one anchor and 5, 10, 20 and 100 agents. (Right) Effect of self-weights on the convergence rate.

the opportunistic nature of the algorithm, as the number of agents increases each agent is more likely to find triangulation sets and perform successful updates. Therefore, the algorithm converges faster as the number of agents increases. In Fig. 4.11 (Right), we study the effect of the self-weights on the convergence rate of the algorithm in networks with one anchor and 5, 10 and 20 agents as we change $\alpha_k = \beta$ from 0.01 to 0.5. It can be seen that the convergence rate decreases as α_k increases. In Fig. 4.12 (Left), we show the percentage of the iterations an agent finds different number of neighbors in networks with one anchor and 5, 10 and 20 agents. For example, an agent

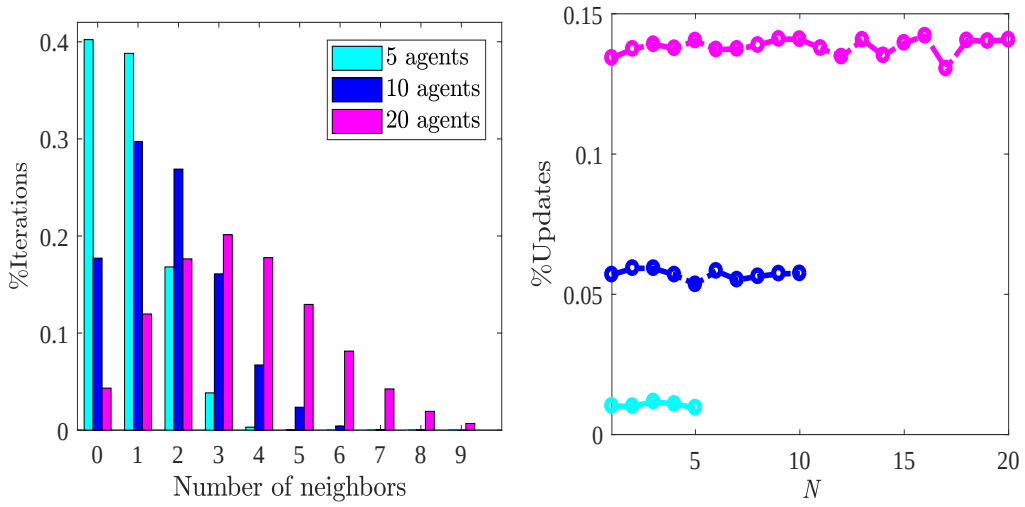

Figure 4.12: (Left) Number of neighbors. (Right) Number of updates/iterations.

finds no neighbors during 40% of the iterations in a network of 5 agents. Fig. 4.12 (Right) shows the ratio of the total number of updates to the number of iterations. We fix the number of iterations to 3000, and take the average over $n = 20$ Monte Carlo simulations. On average, an agent in networks with one anchor and 5, 10, and 20 agents, performs 31, 171, and 422 updates, respectively.

We examine the effect of highly adverse initial conditions in Fig. 4.13 for a network with one anchor and 100 agents. In this figure, thick black curves represent the mean error over all agents. Despite very large initial errors, which is 30 times larger than the dimension of the region, the algorithm converges in less than 500 iterations in this simulation. This convergence rate is slower than the average convergence rate illustrated in Fig. 4.11 (Left) for networks

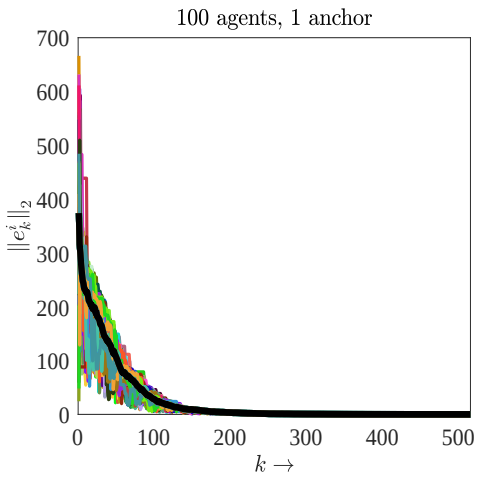

Figure 4.13: Convergence under highly adverse initial estimates.

with 100 agents primarily because of the large initial error.

Localization in the presence of noise

We use two different models to examine the effects of noise on the proposed localization algorithm; First, we assume that the noise on odometry measurements, i.e., the distance and angle that agent i travels at time k, are Gaussian with zero mean and the following variances:

$$\sigma_d^{i\,2} = {K_d}^2 D_k^i, \quad \sigma_\theta^{i\,2} = {K_\theta}^2 D_k^i,$$

where D_k^i represents the total distance that agent i has traveled up to time k. We also assume that the noise on the distance measurement (to a neighboring agent) at time k is normal with zero mean and the variance of

$$\sigma_r^{i\,2} = {K_r}^2 k.$$

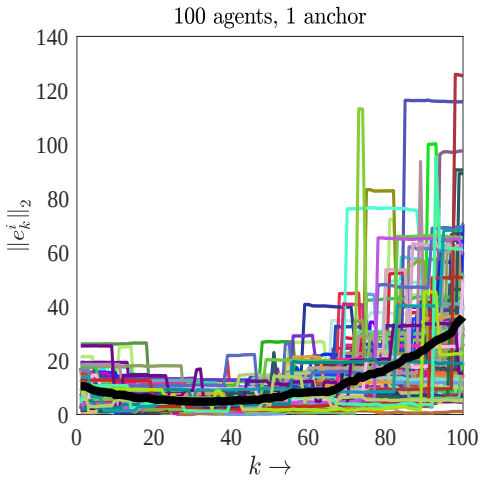

Figure 4.14: Effect of noise on the convergence.

Therefore the variances of the odometry measurements are proportional to the total distance an agent has traveled, and the variance on the distance measurements (to the neighboring agents) increases with time. Such assumptions are common in the relevant literature, e.g., [26, 155–157]. As shown in Fig. 4.14 for a network with one anchor and 100 agents, setting $K_d = K_\theta = K_r = 5 * 10^{-3}$ leads to an unbounded error, which is due to incorrect inclusion test results and the continuous location drifts because of the noise on the distance measurements and the noise on motion, respectively. However, by modifying the algorithm according to Section 4.6, it can be seen in Fig. 4.15 that the localization error is bounded by the communication radius. In the simulations with noise we choose $\epsilon = 20\%$, i.e., an agent performs an update only if the relative inclusion test error, corresponding to the candidate triangulation set

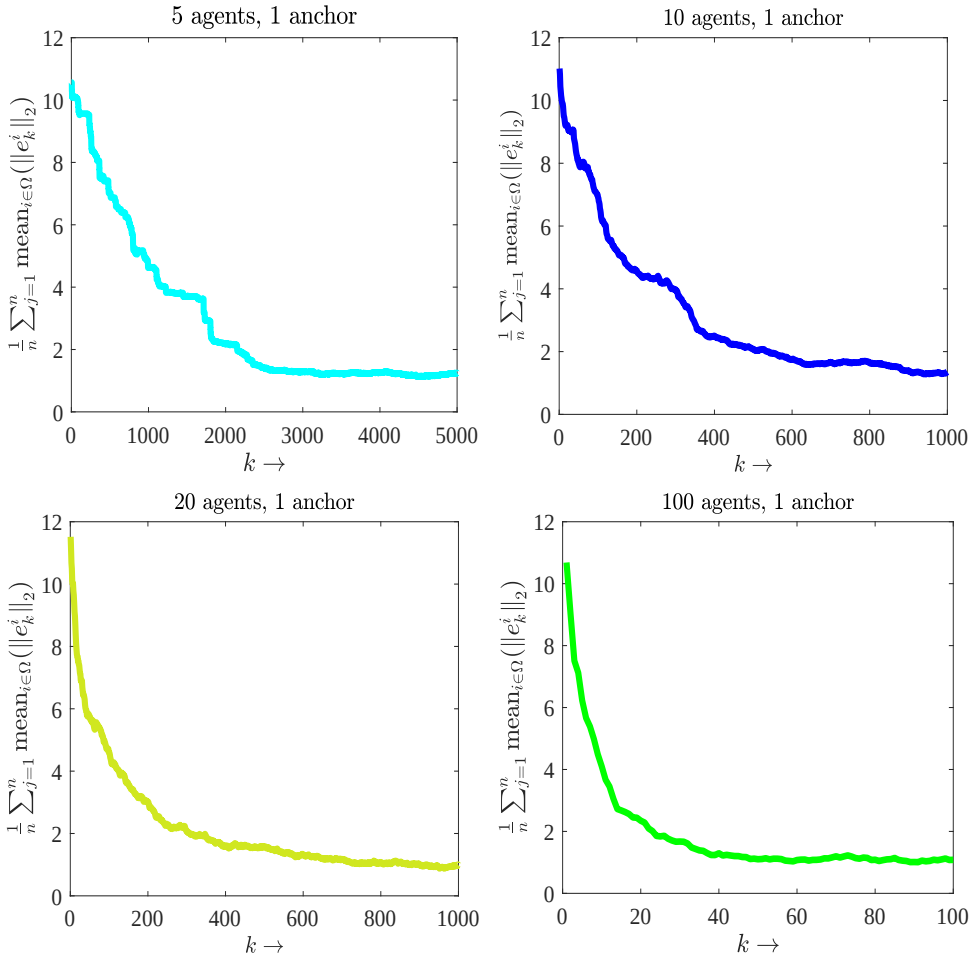

Figure 4.15: Modified algorithm under first noise model.

is less than 20%.

We evaluate the performance of the algorithm on a different noise model, where at each and every iteration the amount of noise on odometry and distance measurements are proportional to the measurements. In Fig. 4.16, we show the simulation results when the amount of noise at each iteration is up to ±5% of the measurements. All the simulations in the presence of noise are

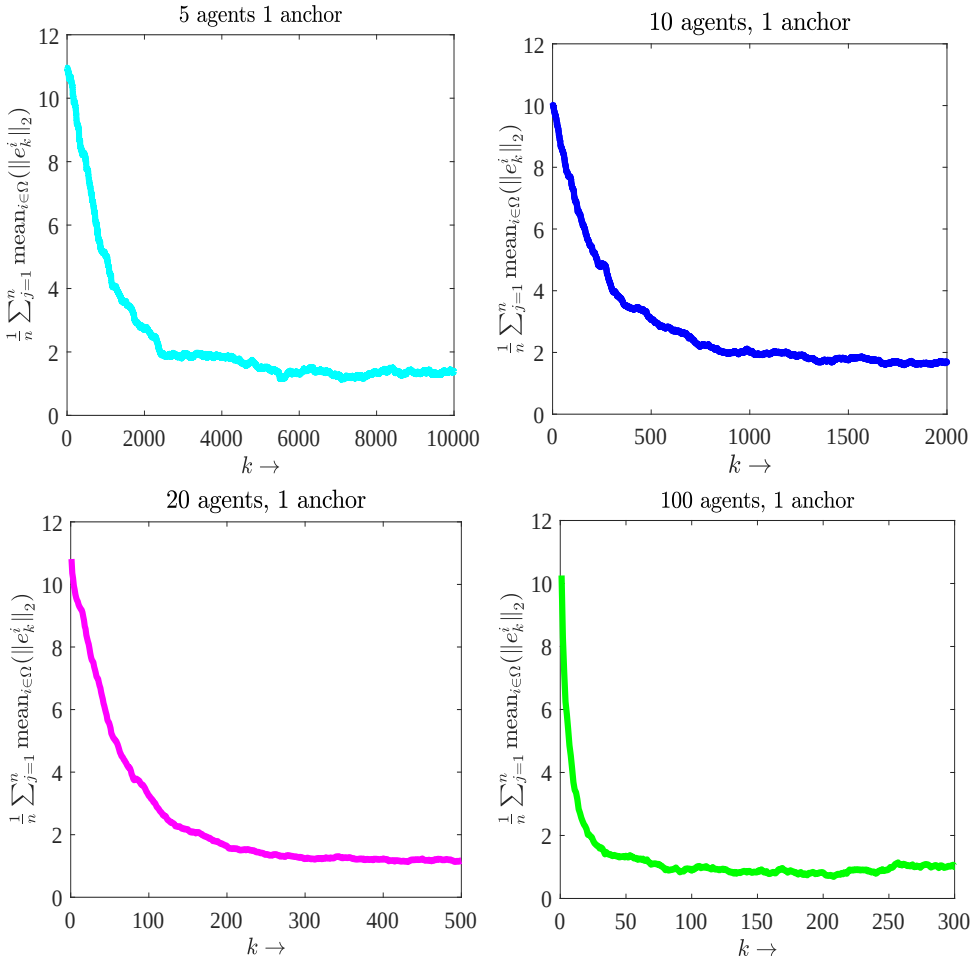

Figure 4.16: Modified algorithm under second noise model.

averaged over $n = 20$ Monte Carlo simulations.

Performance evaluation

We now evaluate the performance of our algorithm in contrast with some well-known localization methods; MCL [36], MSL* [35], and Range-based

SMCL, [37][7]. In Fig. 4.17, we compare the localization error in the Convex

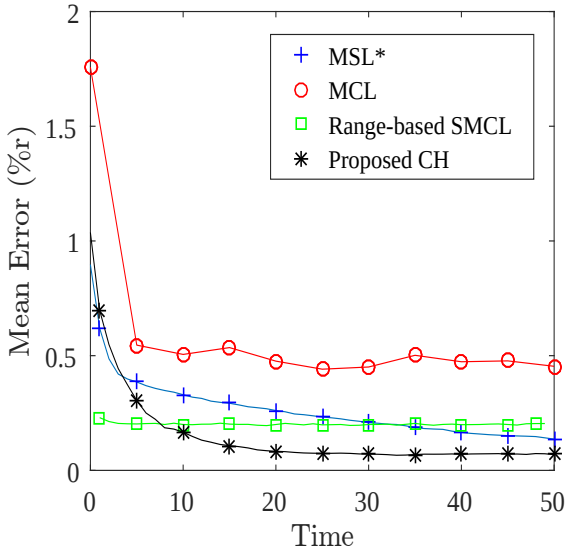

Figure 4.17: Accuracy comparison.

Hull (CH) algorithm with MCL, MSL*, and Range-based SMC. As shown in Table 4.1, in these algorithms node density, n_d, and anchor (seed) density, n_s, denote as the average number of nodes and anchors in the neighborhood of an agent, respectively. Total number of agents and anchors can therefore be determined by knowing these densities as well as the area of the region. We consider $N = 100$ agents and $M = 10$ anchors to remain consistent with the setup in [35–37], and use the same metric, i.e., the location error as a percent-

[7]Monte Carlo Localization (MCL) method exploits mobility to improve the accuracy of localization. Inspired by [36], the authors in [35] propose Mobile and Static sensor network Localization (MSL*) that extends MCL to the case where some or all nodes are static or mobile. On the other hand, Ref [37] provides Range-based Sequential Monte Carlo Localization method (Range-based SMCL), which combines range-free and range-based information to improve localization performance in mobile sensor networks.

age of the communication range. Each data point in Fig. 4.17 is computed by averaging the results of 20 simulation experiments. We keep the other parameters the same as described earlier in this section. With high measurement noise, i.e., in the presence of 10% noise on the range measurements and 1% noise on the motion, our algorithm outperforms MCL, MSL* and Range-based SMCL after 10 iterations. Clearly, the localization error in our algorithm decreases as the amount of noise decreases, and our algorithm converges to the exact agent locations in the absence of noise. Table 4.1 summarizes the performance of the proposed CH algorithm in comparison to the above methods. For a comparison with extended Kalman filter type methods, we refer the

Localization algorithm	**Simulation environment, Network size, Communication range**	**Average error %r**
MCL	500*500, n_d=10, n_s=1, r =50	0.2 - 0.6
MSL*	500*500, n_d=10, n_s=2, r =100	0.1 - 0.5
Range-based SMCL	1*1, n_d=13.9, n_s=1.3, r =0.125	0.2 - 0.4
Proposed CH	20*20, N=100, M=10, r=2	< 0.1

Table 4.1: Comparative performance of localization algorithms.

interested reader to the experiments in [25, 158].

4.9 Remarks

In this section, we shed some light on different aspects of the proposed localization algorithm.

Convergence rate

In some particular applications, e.g., search and rescue in hazardous environments, the agents need not only to find their locations, but also to finish the localization process successfully *in a finite time*. The convergence rate of the algorithm becomes crucial in such applications. Since the algorithm is asymptotic, one can design appropriate termination criteria that are application-dependent. For example, one such criterion can be designed according to the number of iterations typically needed given the size, mobility, models, and noise parameters, as evident from the simulation figures in Section 4.8. We do note that the convergence improves dramatically as the size of the network increases.

Computational complexity

As explained in Appendix B.2, the Cayley-Menger determinant is the determinant of an $(m+2) \times (m+2)$ symmetric matrix that relates the distances among the $m+1$ points in a set, $\Theta_\ell \in \mathbb{R}^m$, to the volume of their convex hull.

Thus, the dimension of such determinants only depends on the dimension of the corresponding Euclidean space. Considering our localization algorithm in $\mathbb{R}^2$, when an agent finds enough (at least $m+1=3$) neighbors, it has to calculate $m+2=4$ Caley-Menger determinants of 4×4 matrices to perform the inclusion test. Similarly, in $\mathbb{R}^3$, when an agent finds at least $m+1=4$ neighbors, it needs to calculate $m+2=5$ determinants of 5×5 matrices to perform the inclusion test. Note that to perform the inclusion test in $\mathbb{R}^2$ and $\mathbb{R}^3$, an agent has to compute four areas and five volumes, respectively. Since the complexity of the computation of an $n\times n$ determinant is $\mathcal{O}(n!)$, considering at most one update per iteration, the computation complexity of the algorithm for an updating agent in $\mathbb{R}^m$ is $(m+2)\mathcal{O}((m+2)!)$, where $m\leq 3$.

Communication loss and drops

In the event of a temporary communication loss/drop, an agent will not be able to perform an update and/or serve as a neighboring node in any triangulation set. However, since the algorithm is opportunistic and fully distributed, this is not going to affect the performance of the rest of the team except that it reduces the chance for the other agents to find a triangulation set. As the size of the network grows, such effect becomes negligible. However, if the failure is permanent, i.e., one agent gets isolated from the network by the end of the

localizing process, it will not be able to find its location and the size of the network gets reduced by one.

Challenges and future work

We note that a successful implementation of this algorithm requires a knowledge of the associated parameters, $\alpha, \alpha^{prime}, \alpha_k, \epsilon$. We provide some insight into these questions in the simulation section of this chapter and have discussed the theoretical relevance of each one throughout the chapter. The effects of changing these parameters on the convergence or performance of the underlying localization algorithm, which may require a large number of practical experiments, can be studied in the future. As a specific example, if the statistics of the distance/motion noise variables are known, one may be able to design ϵ so the inclusion test is correct (with a very high probability) regardless of the noise parameters.

4.10 Summary

In this chapter, we provide a *linear* distributed algorithm to localize an arbitrary number of mobile agents moving in a bounded region. We assume that each agent can measure a noisy version of its motion as well as its distance to the neighboring nodes. We consider an opportunistic algorithm such that the

agents update their location estimates in $\mathbb{R}^m$, as a linear-convex combination of their $m+1$ neighbors if they lie inside their convex hull. The updating agent uses the opportunistic information exchange to refine its location estimate by using a barycentric-based convex update rule. We abstract the algorithm as an LTV system with (sub-) stochastic matrices, and show that it converges to the true agent locations under some mild regularity conditions on update weights. We also relate the dimension of motion in the network to the number of anchors required, and show that a network of mobile agents with full degrees of freedom in the motion can be localized precisely, as long as there is at least one anchor in the network. We evaluate the performance of the algorithm in the presence of noise and provide modifications to the proposed algorithm to address the undesirable effects of noise.

Chapter 5

Localization in dynamic networks via virtual convex hulls

In this chapter, we extend the localization algorithm provided in Chapter 4 by allowing the agents to perform location updates without physically lying inside the convex hull of their neighbors. We consider a network containing *at least one* anchor, and develop an alternative distributed algorithm to localize an arbitrary number of agents moving in a bounded region of interest in $\mathbb{R}^2$. We assume that each agent measures a noisy version of its motion and the distances to the nearby agents.

Recall that in the localization algorithm provided in Section 4.4, a location update is only possible when an agent lies inside a convex hull of m neighbors in $\mathbb{R}^m$ (a triangle of three neighbors in $\mathbb{R}^2$). In this chapter, we introduce the notion of *virtual convex hull*, which allows an agent to perform a location update without passing the inclusion test at any given time. To this aim, we provide a *geometric approach*, which allows each agent to: (i) continually update the distances to the locations where it has exchanged information with the other nodes in the past; and (ii) measure the distance between a neighbor and any such locations. Based on this approach, we provide a linear algorithm to find the locations of an arbitrary number of mobile agents when they follow some convexity in their deployment and motion in $\mathbb{R}^2$. We show that the corresponding localization algorithm, in the absence of noise, can be abstracted as an LTV system, with non-deterministic system matrices. Using the results developed in Chapter 2, we show that this system asymptotically tracks the true locations of the agents.

We now describe the rest of this chapter. In Section 5.1, we formulate the problem. We introduce the geometric approach to track the distances in Section 5.2. We then provide our localization algorithm that relies on the notion of virtual convex hull in Section 5.3, followed by the convergence analysis in Section 5.4. In Section 5.5, we examine the effects of noise on the algorithm. We provide simulation results in Section 5.6 and finally, Section 5.7 concludes

the chapter.

5.1 Problem formulation

Consider a network of N mobile agents, in the set Ω, with unknown locations, and M anchor(s), in the set κ, all located in $\mathbb{R}^2$; let $\Psi = \Omega \cup \kappa$ be the set of all nodes. Let $\mathbf{x}_k^{i*} \in \mathbb{R}^2$ be a row vector that denotes the *true location* of the i-th agent, $i \in \Omega$, at time k, where $k \geq 0$ is the discrete-time index. Regardless of what motion modality is employed by the agents, their motion can be expressed as the deviation between the current and next locations, i.e.,

$$\mathbf{x}_{k+1}^{i*} = \mathbf{x}_k^{i*} + \widetilde{\mathbf{x}}_{k+1}^{i}, \qquad i \in \Psi, \tag{5.1}$$

where $\widetilde{\mathbf{x}}_k^i$ is the true motion vector at time k. We assume that the agents move along straight lines, or otherwise the motion trajectories can be approximated by a piecewise linear model. This is a common assumption in robotics literature, [159], e.g., a simple differential wheeled robot whose wheels run in the same direction and speed, will move in a straight line. We also assume that agent i measures a noisy version, $\widehat{\mathbf{x}}_k^i$, of this motion, e.g., by using an accelerometer:

$$\widehat{\mathbf{x}}_k^i = \widetilde{\mathbf{x}}_k^i + n_k^i, \tag{5.2}$$

where n_k^i is the accelerometer noise at time k. We assume that the motion is restricted in a bounded region in $\mathbb{R}^2$. We define the distance between any two nodes, i and j, measured at the time of communication, k, as $\widetilde{d}_k^{ij}$. We assume that the distance measurement, $\widehat{d}_k^{ij}$, at node i is not perfect and includes noise, i.e.,

$$\widehat{d}_k^{ij} = \widetilde{d}_k^{ij} + r_k^{ij}, \tag{5.3}$$

in which r_k^{ij} is the noise in the distance measurement at time k. The problem is to find the locations of the mobile agents in the set Ω given any initialization of the underlying algorithm.

Main idea: In order to avoid nonlinearity in the solution[1], we consider a barycentric-based linear representation of locations; the advantage of a linear approach is that the convergence is independent of the initial location estimates. Let Θ_k^i denote a set of three agents such that agent i lies inside their convex hull, $\mathcal{C}(\Theta_k^i)$, at time k. In the barycentric-representation, the location of node i is described by a linear-convex combination of the nodes in Θ_k^i, i.e.,

$$\mathbf{x}_k^{i*} = a_k^{i1}\mathbf{x}_k^{1*} + a_k^{i2}\mathbf{x}_k^{2*} + a_k^{i3}\mathbf{x}_k^{3*}, \qquad \Theta_k^i = \{1,2,3\}, \tag{5.4}$$

[1]When only distances to at least $m+1$ anchors are known in $\mathbb{R}^m$, trilateration, e.g., in $\mathbb{R}^2$, requires 3 anchors and solves 3 circle equations, [160]. Clearly, trilateration is nonlinear and coupled in the coordinates; while an iterative procedure built on these nonlinear updates does not converge in general.

in which $\mathbf{x}_k^{i*}$ represents the true location of node i at time k, and the coefficients, a_k^{ij}'s, are the barycentric coordinates, given by

$$a_k^{ij} = \frac{A_{\Theta_k^i \cup \{i\} \setminus j}}{A_{\Theta_k^i}}, \tag{5.5}$$

where $A_{\Theta_k^i \cup \{i\} \setminus j}$ denotes the area of the convex hull formed by the agents in the subscript set. To implement this update, agent i has to lie inside the convex hull of the (three) neighbors in the set Θ_k^i. Appendix B.1 provides a simple procedure to test if an agent lies inside or outside of the convex hull formed by the nearby nodes. Finally, note that the barycentric coordinates are always positive and they sum to 1.

Challenges and our approach: To implement the above barycentric-based procedure, agent i must acquire the mutual distances among the 4 nodes (in the set $\{i \cup \Theta_k^i\}$). However, an agent may never find such a convex hull because the nodes are mobile, in a region possibly full of obstacles, with limited communication and/or visual radius. To address this challenge, we provide a geometric approach that allows an agent to track its distance to any location, where it has exchanged information with the other nodes in the past (Section 5.2). We further show that at the time of communication an agent can compute the distance between a neighbor and any such locations (Section 5.2). Using these methods, we introduce the notion of a virtual convex hull, which is a triangle whose vertices are located at the virtual locations

where the agent exchanged information in the past (Section 5.3). The described approach results in an algorithm where an update occurs only when a certain set of real and virtual conditions are satisfied. The subsequent analysis is to characterize the convergence of such LTV algorithms whose system matrices are non-deterministic. We now enlist our assumptions:

B0: Each anchor, $i \in \kappa$, knows its location, $\mathbf{x}_k^{i*} = \mathbf{x}_0^{i*}$, $\forall k$.

B1: Each agent, $i \in \Omega$, has a noisy measurement, $\widehat{\mathbf{x}}_k^i$, of its motion vector, $\widetilde{\mathbf{x}}_k^i$, see Eq. (5.2).

B2: Each agent, $i \in \Omega$, has a noisy measurement, $\widehat{d}_k^{ij}$, of the distances to nodes, $j \in \Psi$ within a radius, r, see Eq. (5.3).

Under the above assumptions, we are interested in finding the true locations of each agent in the set Ω, without the presence of any central coordinator. In the following, we first discuss the ideal scenario when the measurements are not affected by noise. We then evaluate the performance of the algorithm in the presence of noise, and present a modified algorithm to counter the undesirable effects of noise (Section 5.5).

5.2 A geometric approach towards localization

In this section, we consider the motion and distance in the noiseless case, i.e., $n_k^i = 0$ and $r_k^{ij} = 0$, in Eqs. (5.2) and (5.3). Let the *true* location of the i-th agent, $i \in \Omega$ in $\mathbb{R}^2$, be decomposed as

$$\mathbf{x}_k^{i*} = [x_k^{i*} \; y_k^{i*}] \qquad \forall k \geq 0.$$

We describe the motion of agent i as:

$$\begin{aligned} x_{k+1}^{i*} &= x_k^{i*} + d_{k\to k+1}^i \cos\left(\sum_{k'=1}^{k+1} \theta_{k'\to k'+1}^i + \theta_0^i\right), \\ y_{k+1}^{i*} &= y_k^{i*} + d_{k\to k+1}^i \sin\left(\sum_{k'=1}^{k+1} \theta_{k'\to k'+1}^i + \theta_0^i\right), \end{aligned} \tag{5.6}$$

where $d_{k\to k+1}^i$ and $\theta_{k\to k+1}^i$ denote the distance and angle traveled by agent i, between time k and $k+1$, and θ_0^i is agent i's initial orientation[2], see Fig. 5.1. Let us also consider a vision-based distance measurement process, where an agent needs to see another agent with its camera in order to find the mutual distance, as opposed to wireless-based distance measurements that are prone to larger errors. When agent i moves to a new location at time k, it first scans the neighborhood, i.e., rotates by an angle, $0 \leq \beta_k^i < 2\pi$, in order to find (and make visual contact with) another node (agent or anchor). If the agent does

[2]Note that these initial orientations, θ_0^i's, are arbitrary and we neither assume any global synchronization nor we know what the true angles are.

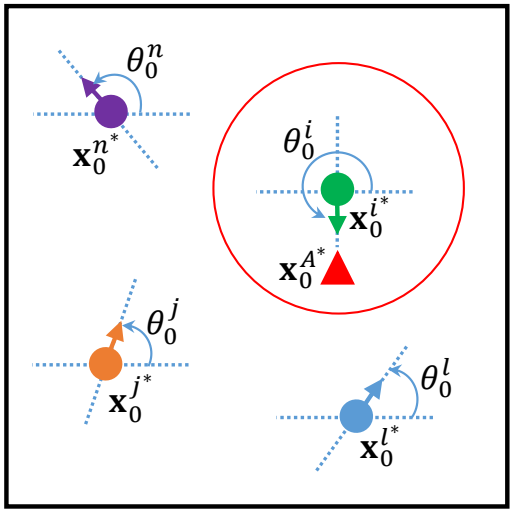

Figure 5.1: Initial orientations of 4 agents.

not find any neighbor at time k, then $\beta_k^i = 0$. If on the other hand, agent i finds another node within its communication radius, it exchanges information with that node, then makes another rotation by an angle, $0 \leq \alpha_k^i < 2\pi$, which is randomly chosen at each iteration, and travels the distance of $d_{k\to k+1}^i$ in the new direction. Thus, the angle traveled by agent i between time k and $k+1$, $\theta_{k\to k+1}^i$, can be represented as

$$\theta_{k\to k+1}^i = \alpha_k^i + \beta_k^i, \qquad k \geq 0. \tag{5.7}$$

Before we proceed, note that under perfect motion and noiseless distance measurements, an agent cannot find its location by direct communication with an anchor, unless it knows its angle towards the anchor with respect to a mutual frame; in contrast, the proposed framework in this chapter assumes that *only relative angles* are known to each agent. To clarify this, consider Fig. 5.1,

in which filled colored circles and red triangle indicate the initial locations of the agents, i, j, l, n and the anchor, A, respectively, and red circle shows the communication radius of agent i. Agent i in Fig. 5.1 initially finds anchor A in its communication radius. However, in order to find its exact location, agent i needs to know θ_0^i, in addition to $\mathbf{x}_0^{A*}$ and $\widetilde{d}_0^{iA}$. We assume that such angles are not available to the agents.

In what follows, we first explain the procedure to track the distance between an agent and the locations where it has exchanged information with other nodes in the past. We then show how an agent can compute the distance between a neighbor and any such location at the time of communication.

Tracking the distance after a direct communication

Consider an agent, $i \in \Omega$, to fall within a distance, r, of node, $j \in \Psi$, at some time k_j; by Assumption **B2**, agent i can measure its distance, $\widetilde{d}_{k_j}^{ij}$ to node j, and receive j-th node's location estimate, hereinafter denoted as $\mathbf{x}_{k_j}^j$. Once this information is acquired at time k_j, agent i tracks the distance to the *true location*, $\mathbf{x}_{k_j}^{j*}$, for all $k \geq k_j$, even when the two agents move apart. In other words, agent i, has the following information for each node it has communicated with:

$$\{j, k_j, \widetilde{d}_k^{ij}, \mathbf{x}_{k_j}^j\}, \qquad k \geq k_j, j \in \Psi, \tag{5.8}$$

where k_j is the instant of the most-recent contact, and $\widetilde{d}_k^{ij}$ is the distance between true locations, $\mathbf{x}_k^{i*}$ and $\mathbf{x}_{k_j}^{j*}, k \geq k_j$. This procedure is illustrated in Fig. 5.2, in which agent i is indicated by red filled circle, agents j and l are represented by blue and green circles, respectively, and red circle indicates the communication radius. Note that in order to make contact with agent j, agent

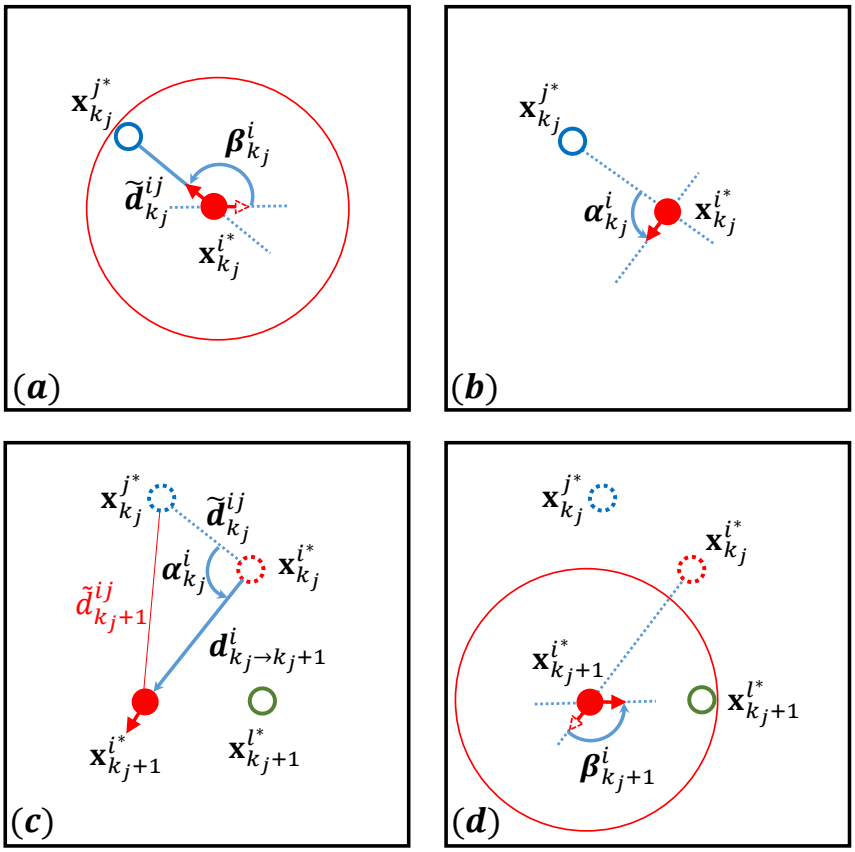

Figure 5.2: Distance tracking after a direct communication.

i has changed its orientation by $\beta_{k_j}^i$ at time k_j, see Fig. 5.2 (a). After the two agents exchange information, agent i changes its orientation by $\alpha_{k_j}^i$, Fig. 5.2 (b), and travels the distance of $d_{k_j\to k_j+1}^i$ in the new direction, Fig. 5.2 (c). We

now show how an agent tracks the distance to a virtual location, where it has exchanged information with another node in the past.

Lemma 9. *Consider agent, $i \in \Omega$, with true location, $\mathbf{x}^{i*}_{k_j}$. Suppose agent i communicates with node, $j \in \Psi$, at time k_j. Let $\widetilde{d}^{ij}_{k_j}$ be the distance between i and j at the time of communication. Suppose agent i and j move apart at time $k_j + 1$, where the motion is given by $d^i_{k_j \to k_j+1}$ and $\theta^i_{k_j \to k_j+1}$, both known to agent i. The distance between the true locations, $\mathbf{x}^{i*}_{k_j+1}$ and $\mathbf{x}^{j*}_{k_j}$, is*

$$\widetilde{d}^{ij}_{k_j+1} = \sqrt{(\widetilde{d}^{ij}_{k_j})^2 + (d^i_{k_j \to k_j+1})^2 - 2\widetilde{d}^{ij}_{k_j} d^i_{k_j \to k_j+1} \cos(\alpha^i_{k_j})}. \tag{5.9}$$

Proof. Consider the triangle, whose vertices are at $\mathbf{x}^{i*}_{k_j}$, $\mathbf{x}^{i*}_{k_j+1}$ and $\mathbf{x}^{j*}_{k_j}$. To find the current distance of agent i, from the location of agent j at time k_j, we can use the *law of cosines*, which connects the length of an unknown side of a triangle to the lengths of the two other sides and the angle opposite to the unknown side. This triangle is depicted in Fig. 5.2 (c), in which the two known side lengths are $\widetilde{d}^{ij}_{k_j}$ and $d^i_{k_j \to k_j+1}$. Thus, knowing the angle between these sides of the triangle, $\alpha^i_{k_j \to k_j+1}$, the length of the third side can be determined according to the following

$$(\widetilde{d}^{ij}_{k_j+1})^2 = (\widetilde{d}^{ij}_{k_j})^2 + (d^i_{k_j \to k_j+1})^2 - 2\widetilde{d}^{ij}_{k_j} d^i_{k_j \to k_j+1} \cos(\alpha^i_{k_j}),$$

which corresponds to Eq. (5.9), and completes the proof. □

Note that at time $k_j + 1$, agent i finds agent l in its communication radius,

hence changes its direction by $\beta^i_{k_j+1}$ in order to make contact with agent l, see Fig 5.2 (d).

Finding the distance with indirect communication

We now show how agent i can use the distance/angle information to find the distances between a neighbor and any virtual location, where it has previously communicated with another node.

Lemma 10. *Suppose agent i has previously made contact with node j, hence possesses the following information:*

$$\{j, k_j, \widetilde{d}^{ij}_k, \mathbf{x}^j_{k_j}\}, \qquad k \geq k_j.$$

Suppose agent i finds a neighbor, say agent ℓ, at time $k_\ell > k_j$. Agent i can then find the distance between $\mathbf{x}^{j*}_{k_j}$ and $\mathbf{x}^{\ell*}_{k_\ell}$ as follows:

$$\widetilde{d}^{j\ell}_{k_\ell} = \sqrt{(\widetilde{d}^{ij}_{k_\ell})^2 + (\widetilde{d}^{i\ell}_{k_\ell})^2 - 2\widetilde{d}^{ij}_{k_\ell}\widetilde{d}^{i\ell}_{k_\ell}\cos\angle(\widetilde{d}^{ij}_{k_\ell}, \widetilde{d}^{i\ell}_{k_\ell})}, \tag{5.10}$$

in which $\widetilde{d}^{ij}_{k_\ell}$ is the distance between $\mathbf{x}^{i*}_{k_\ell}$ and $\mathbf{x}^{j*}_{k_j}$, and $\angle(\widetilde{d}^{ij}_{k_\ell}, \widetilde{d}^{i\ell}_{k_\ell})$ is the angle between the two lines connecting $\mathbf{x}^{i*}_{k_\ell}$ to $\mathbf{x}^{j*}_{k_j}$ and $\mathbf{x}^{\ell*}_{k_\ell}$.

Proof. We illustrate this procedure in Fig. 5.3, where all known/measured distances and angles are distinguished from the unknowns by bold lines and bold arcs, respectively. After agent i makes contact and exchange information with node j at time k_j, it moves the distance of $d^i_{k_j \to k_j+1}$ to the new location, $\mathbf{x}^{i*}_{k_j+1}$.

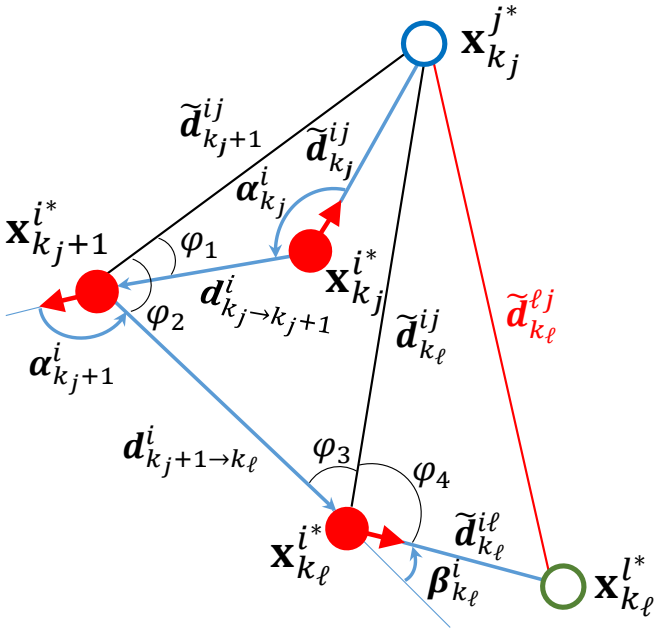

Figure 5.3: At time k_ℓ, agent i finds the distance between a new neighbor, node ℓ, and the virtual location, $\mathbf{x}^{j*}_{k_j}$, where it exchanged information with node j at time $k_j < k_\ell$.

At this point, agent i can (use the law of cosines to) find $\widetilde{d}^{ij}_{k_j+1}$ as a function of $\widetilde{d}^{ij}_{k_j}$, $d^{i}_{k_j\to k_j+1}$, and $\alpha^{i}_{k_j}$, which are all known to this agent, i.e.,

$$\widetilde{d}^{ij}_{k_j+1} = f(\widetilde{d}^{ij}_{k_j}, d^{i}_{k_j\to k_j+1}, \alpha^{i}_{k_j}).$$

Knowing $\widetilde{d}^{ij}_{k_j+1}$, agent i can then find

$$\phi_1 = g(\widetilde{d}^{ij}_{k_j}, d^{i}_{k_j\to k_j+1}, \widetilde{d}^{ij}_{k_j+1}).$$

Note that given the three sides of a triangle, each angle can be computed. Since agent i cannot find any neighbor at time k_j+1, we have $\beta^{i}_{k_j+1} = 2\pi$. Agent i then changes its direction by $\alpha^{i}_{k_j+1}$ and travels the distance of $d^{i}_{k_j+1\to k_\ell}$

to the new location, $\mathbf{x}_{k_\ell}^{i*}$, where $k_\ell = k_j + 2$. At this point, agent i can find

$$\widetilde{d}_{k_\ell}^{ij} = f(\widetilde{d}_{k_j+1}^{ij}, d_{k_j+1\to k_\ell}^{i}, \phi_2),$$

in which $\phi_2 = \pi - \alpha_{k_j+1}^{i} + \phi_1$. Agent i can then find

$$\phi_3 = g(\widetilde{d}_{k_\ell}^{ij}, d_{k_j+1\to k_\ell}^{i}, \widetilde{d}_{k_j+1}^{ij}),$$

at time k_ℓ. Since agent i finds node ℓ within the communication radius at time k_ℓ, it changes its direction by $\beta_{k_\ell}^{i}$ in order to make a contact with agent ℓ. Finally, knowing $\widetilde{d}_{k_\ell}^{i\ell}$, $\widetilde{d}_{k_\ell}^{ij}$, and, $\phi_4 = \angle(\widetilde{d}_{k_\ell}^{ij}, \widetilde{d}_{k_\ell}^{i\ell}) = \pi - \phi_3 - \beta_{k_\ell}^{i}$, agent i can find the distance between $\mathbf{x}_{k_j}^{j*}$ and $\mathbf{x}_{k_\ell}^{\ell*}$ according to Eq. (5.10). □

In the next section, we use these distance tracking methods to describe the localization algorithm in $\mathbb{R}^2$.

5.3 Localization algorithm

Consider a network of N agents with unknown locations and M anchors, according to the motion model introduced in Section 5.1. Let $\mathcal{V}_k^i \subseteq \Psi$ be the *i-visited set*, defined as the set of distinct nodes visited by agent, $i \in \Omega$, up to time k; and call an element in this set as *i-visited node*. We start by introducing the notion of a virtual convex hull.

Virtual convex hull

Suppose agent i communicates with node j at time k_j, and obtains the distance, $\widetilde{d}^{ij}_{k_j}$ to j, along with j's current location estimate, $\mathbf{x}^j_{k_j}$, i.e., $j \in \mathcal{V}^i_{k_j}$. At any time, $k > k_j$, agents, i and j, may move apart but agent i now knows $\widetilde{d}^{ij}_k$, $\forall k > k_j$, using the geometric framework discussed in Section 5.2. At some later time, $k_\ell > k_j$, agent i makes contact with another node, ℓ, and thus obtains $\mathbf{x}^\ell_{k_\ell}$ and $\widetilde{d}^{i\ell}_{k_\ell}$, and keeps track of $\widetilde{d}^{i\ell}_k$, $\forall k > k_\ell$, thus $j, \ell \in \mathcal{V}^i_k, \forall k \geq k_\ell$. Using the approach described in Section 5.2, at time k_ℓ, agent i also computes $\widetilde{d}^{j\ell}_{k_\ell}$, i.e., the distance between $\mathbf{x}^{j*}_{k_j}$ and $\mathbf{x}^{\ell*}_{k_\ell}$. Finally, agent i meets agent n at some $k_n > k_\ell$, and thus now possesses the location estimates: $\mathbf{x}^j_{k_j}, \mathbf{x}^\ell_{k_\ell}, \mathbf{x}^n_{k_n}$; and the distances: $\widetilde{d}^{iq}_k, k > k_n$, with $q = j, \ell, n$ (computed by Lemma 9), along with the following distances: $\widetilde{d}^{j\ell}_{k_n}$, $\widetilde{d}^{nj}_{k_n}$, and $\widetilde{d}^{n\ell}_{k_n}$ (computed by Lemma 10). At this point $j, \ell, n \in \mathcal{V}^i_k, \forall k \geq k_n$, and agent i can use the 6 distances to perform the inclusion test described in Appendix B.1 to check if the three visited nodes forms a virtual convex hull in which agent i lies at time k.

If the test is passed, the set, $\Theta^i_k \triangleq \{j, \ell, n\} \subseteq \mathcal{V}^i_k$, forms the virtual convex hull; otherwise, agent i continues to move *and* add nodes in $\mathcal{V}^i_k$ until some combination passes the convexity test. Note that the distance between the agent and each of the virtual locations is updated every time agent i moves to a new location. However, the *distance between the virtual locations* does not

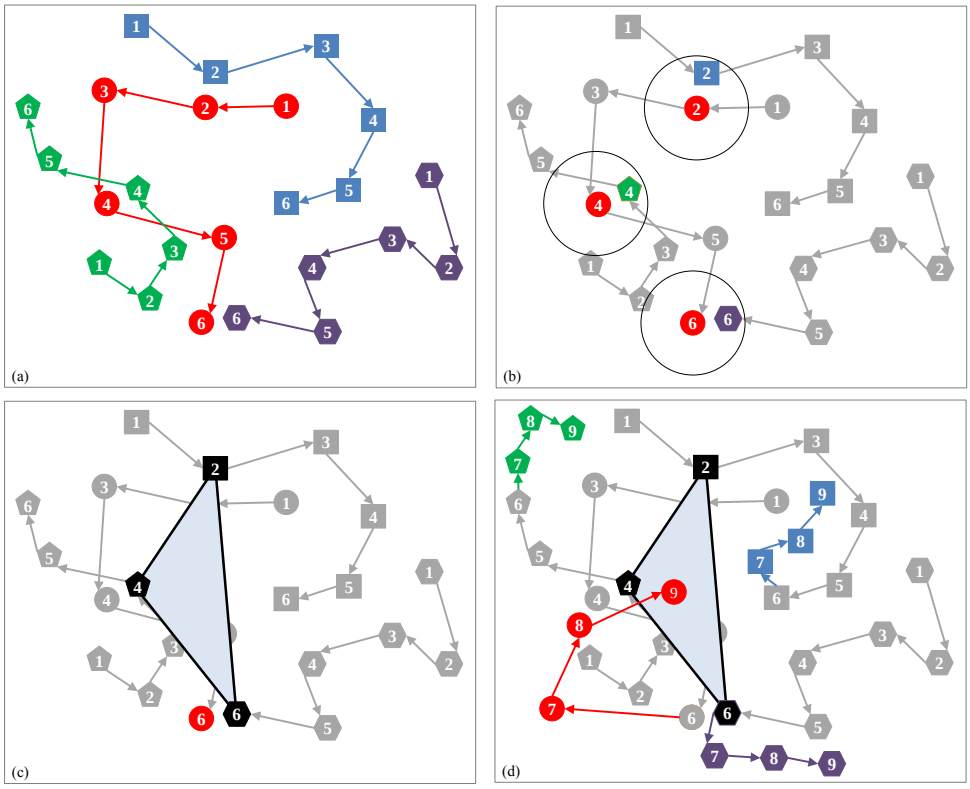

Figure 5.4: Virtual convex hull with four agents: ○, □, ⬠, ⬡; with respect to agent ○: (a) Agent trajectories and time-indices. (b) ○ ↔ □ at $k_{\square} = 4$, ○ ↔ ⬠ at $k_{\pentagon} = 4$, ○ ↔ ⬡ at $k_{\hexagon} = 6$; circles indicate communication radius of agent ○. (c) Virtual convex hull of agents, □, ⬠, ⬡, available at agent ○ at $k = 6$. (d) Trajectories at $k > 6$, test passed at $k = 9$.

change unless agent i *revisits* any of the previously visited nodes.

Fig. 5.4 (a) shows the trajectories of four agents: ○, □, ⬠, ⬡, over $k = 1, \ldots, 9$; the time-indices are marked inside the agent symbols. From the perspective of agent ○, see Fig. 5.4 (b): it first makes contact (communicates) with agent □, at time $k_{\square} = 2$, and then they both move apart; next, it makes contact with agents, ⬠ at $k_{\pentagon} = 4$, and ⬡ at $k_{\hexagon} = 6$. We have $\mathcal{V}_2^{\bigcirc} = \{\square\}$, $\mathcal{V}_4^{\bigcirc} = \{\square, \pentagon\}$, and $\mathcal{V}_6^{\bigcirc} = \{\square, \pentagon, \hexagon\}$, where a non-trivial con-

vex hull becomes available at $k = 6$. However, agent $\bigcirc$ does not lie in the corresponding convex hull, $\mathcal{C}(\mathcal{V}_6^{\bigcirc})$, and cannot update its location estimate with the past neighboring estimates: $\mathbf{x}_{k_{\square}}^{\square}, \mathbf{x}_{k_{\pentagon}}^{\pentagon}, \mathbf{x}_{k_{\hexagon}}^{\hexagon}$. At this point, agent $\bigcirc$ must wait until it *either* moves inside the convex hull of $\square, \pentagon, \hexagon$, *or* finds another agent with which the convexity condition is satisfied. Fig. 5.4 illustrates this process in four frames. The former is shown in Fig. 5.4 (d), where agent $\bigcirc$ has moved inside $\mathcal{C}(\mathcal{V}_6^{\bigcirc})$ at some later time, $k = 9$; we have $\Theta_9^{\bigcirc} = \{\square, \pentagon, \hexagon\}$. The notion of a *virtual* convex hull is evident from this discussion: an agent may only communicate with at most one agent at any given time; when the convexity condition is satisfied eventually, the updating agent may not be in communication with the corresponding nodes. Once $\Theta_k^{\bigcirc}$ is successfully formed, agent $\bigcirc$ updates its location *linearly* using the barycentric representation in Eq. (5.4), where the coefficients are computed using the distance equations in Lemmas 9, 10, and the Cayley-Menger determinant to compute areas. After this update, agent $\bigcirc$ removes $\Theta_k^{\bigcirc}$ from $\mathcal{V}_{\bigcirc}^i$, as the location estimates of the nodes in $\Theta_k^{\bigcirc}$ have been consumed[3].

The following result will be useful in the sequel.

Lemma 11. *For each $i \in \Omega$, there exists a set, $\Theta_k^i \subseteq \mathcal{V}_k^i$, such that $|\Theta_k^i| = 3$ and $i \in \mathcal{C}(\Theta_k^i)$, for infinitely many k's.*

[3]We choose this simple strategy to remove information from $\mathcal{V}_k^i$ for convenience. Another candidate strategy is to use a *forgetting factor*, which chooses the past used nodes less frequently.

Proof. Consider a network of 4 agents, $\{i, j, m, l\}$, and let us focus on finding a triangulation set for agent i. Suppose all agents are moving in an n by n

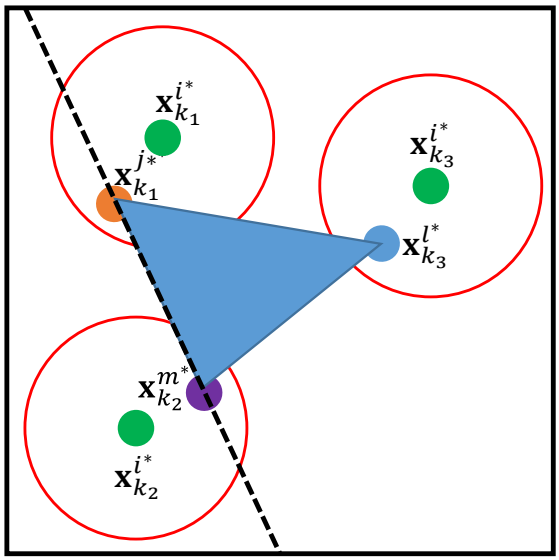

Figure 5.5: Existence of a virtual convex hull.

region, and the communication radius for each agent is r. In order for agent i to communicate with another agent, say j, at time k_1, agent j has to lie inside a circle with radius r centered at agent i's location, $\mathbf{x}_{k_1}^{i*}$, see Fig. 5.5. The probability of such event at time k_1, is therefore given by[4]:

$$\mathbb{P}(j \in \mathcal{V}_{k_1}^i) = \frac{\pi r^2}{n^2}.$$

Similarly, the probability that agent i communicates with agent m at a later

[4]This expression assumes that agents are uniformly distributed over the region at any given time. This assumption can be justified by considering random initial deployment and motion. However, the proof follows as long as there is a nonzero probability for each agent to communicate with other agent(s) in the network.

time, $k_2 > k_1$, and with agent l, at time $k_3 > k_2 > k_1$ can be given by:

$$\begin{aligned} \mathbb{P}(m \in \mathcal{V}^i_{k_2}) &= \frac{\pi r^2}{n^2}, \quad \text{and} \\ \mathbb{P}(l \in \mathcal{V}^i_{k_3}) &= \frac{\pi r^2}{n^2}, \end{aligned}$$

respectively. Note that the probability of agent m being exactly at $\mathbf{x}^{j*}_{k_1}$ at time k_2 is zero, and therefore we can draw a virtual line (the dotted line in Fig. 5.5) between the locations at which agent i has communicated with agents j and m, i.e., between $\mathbf{x}^{j*}_{k_1}$ and $\mathbf{x}^{m*}_{k_2}$. Similarly the probability of agent l lying on the aforementioned line at time k_3 is zero. Thus, the locations where agent i can meet the other three agents at different times with nonzero probability, form a virtual convex hull. Therefore, the three agents j, m and l can pass the inclusion test for agent i at any given time $k > k_3$, if agent i lies inside the corresponding virtual triangle. The probability of such event at any given time, $k > k_3$, is given by

$$\mathbb{P}(i \in \mathcal{C}(\mathbf{x}^{j*}_{k_1}, \mathbf{x}^{m*}_{k_2}, \mathbf{x}^{l*}_{k_3})) = \frac{\text{area}(\triangle(\mathbf{x}^{j*}_{k_1}, \mathbf{x}^{m*}_{k_2}, \mathbf{x}^{l*}_{k_3})}{n^2},$$

which gives a non-zero probability for agent i to find a triangulation set. □

Algorithm

We now describe the localization algorithm according to the number, $|\mathcal{V}^i_k|$, of i-visited nodes in the i-visited set, $\mathcal{V}^i_k$. Main steps of the algorithm are

Algorithm 2 Localize N agents in $\mathbb{R}^2$ in the presence of M anchors with precision p

Require: $M \geq 1$ **and** $N + M \geq 4$
  $k \leftarrow 0$
  $\mathbf{x}_0 \leftarrow$ random initial coordinates
  for $i = 1$ to N **do**
    $\mathcal{V}_0^i = \emptyset$
  end for
  while $k <$ termination criterion[5] **do**
    $k \leftarrow k + 1$
    for $i = 1$ to N **do**
      $\mathcal{V}_k^i \leftarrow$ nodes in the communication radius of agent i at time k
      if $0 \leq |\mathcal{V}_k^i| < 3$ **then**
        do not update
      else
        perform inclusion test on (all possible combinations of) 3 neighbors
        if no triangulation set found **then**
          do not update
        else
          update location according to Eq. (5.11)
        end if
      end if
    end for
  end while

summarized in Algorithm 2. There are two different update scenarios for any arbitrary agent, say i:

(i) $0 \leq |\mathcal{V}_k^i| < 3$: agent i, does not update its current location estimate.

(ii) $|\mathcal{V}_k^i| \geq 3$: agent i, performs the inclusion test; if the test is passed the location update is applied.

[5]The termination criterion can be designed according to the number of iterations typically needed given the size, mobility, models, and noise parameters, as evident from the simulation figures in Section 5.6.

Using the above, consider the following update:

$$\mathbf{x}_{k+1}^i = \alpha_k \mathbf{x}_k^i + (1-\alpha_k) \sum_{j \in \Theta_k^i} a_k^{ij} \mathbf{x}_k^j + \widetilde{\mathbf{x}}_{k+1}^i, \tag{5.11}$$

where $\mathbf{x}_k^i$ is the vector of the i-th agent's coordinates at time k, $\widetilde{\mathbf{x}}_{k+1}^i$ is the motion vector, a_k^{ij} is the barycentric coordinate of node i with respect to the nodes $j \in \Theta_k^i$, and α_k is such that

$$\alpha_k = \begin{cases} 1, & \forall k \mid \Theta_k^i = \emptyset, \\ \in [\beta, 1), & \forall k \mid \Theta_k^i \neq \emptyset, \end{cases} \tag{5.12}$$

where β is a design parameter and Θ_k^i is a virtual convex hull. An updating agent receives the valuable location information only if it updates with respect to an anchor, or another agent that has previously communicated with an anchor. The non-zero self-weights assigned to the previous state of the updating agent guarantees that the agent does not completely forget the valuable information after receiving them, e.g., by performing an update where none of the agents in the triangulation set has previously received anchor information. Note that $\Theta_k^i = \emptyset$ does not necessarily imply that agent i has no neighbors at time k, but only that no set of neighbors meet the (virtual) convexity. The above algorithm can be written in matrix form as

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \widetilde{\mathbf{x}}_{k+1}, \qquad k > 0, \tag{5.13}$$

where $\mathbf{x}_k$ is the vector of agent coordinates evaluated at time k, $\mathbf{u}_k$ is the vector of anchor coordinates at time k, and $\widetilde{\mathbf{x}}_{k+1}$ is the change in the location

of agents at the beginning of the $k-$th iteration according to the motion model. Also $\mathbf{P}_k$ and $\mathbf{B}_k$ are the system matrix and the input matrix of the above LTV system. We denote the (i,j)-th element of the matrices, $\mathbf{P}_k$ and $\mathbf{B}_k$, as $(\mathbf{P}_k)_{i,j}$ and $(\mathbf{B}_k)_{i,j}$, respectively. We can now rewrite Eq. (5.11) as

$$\begin{aligned} \mathbf{x}_{k+1}^i &= \alpha_k \mathbf{x}_k^i + (1-\alpha_k)\left(\sum_{j\in\Theta_k^i\cap\Omega} a_k^{ij}\mathbf{x}_k^j\right), \\ &+ (1-\alpha_k)\left(\sum_{m\in\Theta_k^i\cap\kappa} a_k^{im}\mathbf{u}_k^m\right) + \widetilde{\mathbf{x}}_{k+1}^i, \end{aligned} \tag{5.14}$$

where $\mathbf{u}_k^m$ denotes the known coordinates of the m-th anchor at time k, and

$$a_k^{ij} = \begin{cases} p_k^{ij}, & \text{if } j \in \Theta_k^i \cap \Omega, \\ \\ b_k^{im}, & \text{if } m \in \Theta_k^i \cap \kappa. \end{cases} \tag{5.15}$$

If the triangulation set for agent i at time k does not contain the j-th agent or the m-th anchor, the corresponding elements in system and input matrices, $(\mathbf{P}_k)_{i,j}$ and $(\mathbf{B}_k)_{i,m}$ are zero. Note that Eq. (5.12) immediately implies that the self-weight at each agent is always lower bounded, i.e.,

$$0 < \beta \leq (\mathbf{P}_k)_{i,i} \leq 1, \qquad \forall k, i \in \Omega. \tag{5.16}$$

Guaranteed anchor distribution: Since accurate information is only injected via the anchors, it is reasonable (and necessary) to set a lower bound on the weights assigned to the anchor states. In particular, we make the following assumption.

B3: For any update that involves an anchor, i.e., for any $(\mathbf{B}_k)_{i,m} \neq 0$, we assume that

$$0 < \alpha \leq (\mathbf{B}_k)_{i,m} \neq 0, \qquad \forall k, i \in \Omega, m \in \Theta_k^i \cap \kappa, \tag{5.17}$$

where α is the minimum anchor contribution.

Assumption **B3** implies that if there is an anchor in the (virtual) convex hull, it always contributes a certain amount of information, i.e., the weight assigned to the anchor (which comes from barycentric coordinates) should be at least α. Therefore, the area (in $\mathbb{R}^2$) of the triangle corresponding to the anchor must take an adequate portion of the area of the whole convex hull triangle.

Minimum agent contribution When there is no anchor in the triangulation set, we restrict an agent from performing an update, unless it is located in a proper position inside a candidate (virtual) convex hull. We make the following assumption:

B4: For any agent, $j \in \Omega$, involved in an update, i.e., for any $(\mathbf{P}_k)_{i,j} \neq 0$, we assume that

$$0 < \alpha' \leq (\mathbf{P}_k)_{i,j}, \qquad \forall k, i \in \Omega, j \in \Theta_k^i \cap \Omega, \tag{5.18}$$

where i is the updating agent's index, and α' is the minimum agent contribution.

In Fig. 5.6, we illustrate minimum agent contribution, where we assume $\alpha_k = 0$ for the sake or argument; For agent i to perform an update, the area of the corresponding inner triangle has to constitute a minimum proportion, α', of the area of the convex hull triangle. This is shown in Fig. 5.6 (Left); At time

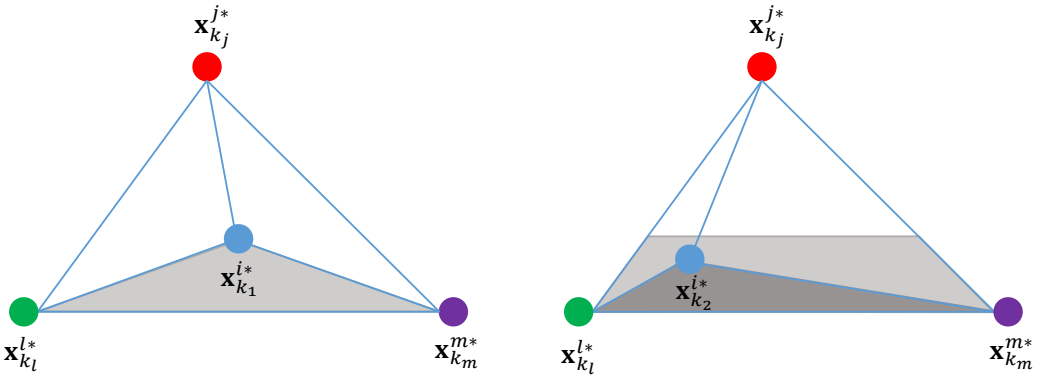

Figure 5.6: Minimum agent contribution.

k_1, agent i is located in the virtual convex hull of the nodes j, m and ℓ, such that the ratio of the area of the shaded triangle to the area of the convex hull triangle is α'. Agent i can perform an update at time k_1. The upper side of the shaded trapezoid in Fig. 5.6 (Right) is the threshold boundary. If agent i remains in the same virtual convex hull, but moves inside the shaded trapezoid, e.g., to $\mathbf{x}_{k_2}^{i*}$ at time k_2, Eq. (4.14) will not hold and no update will occur at that time.

Assumption **B4** can be justified by considering the effects of noise on proposed localization algorithm. If an agent is located close to the boundaries of a (virtual) convex hull, noise on distance measurements may lead to incorrect

inclusion test results, i.e., a set of visited neighbors may fail the convexity test while the agent is indeed located inside the convex hull, or vice versa. To avoid such scenarios, we generalize Assumption **B3** to non-anchor agents.

With the lower bounds on the self-weights according to Eq. (5.16), and the lower bounds on the weights assigned to the anchors and agents according to Eqs. (5.17) and (5.18), we note that at time k, the matrix of barycentric coordinates with respect to agents with unknown locations, i.e., the system matrix $\mathbf{P}_k$, is either

(i) ***identity***, when no update occurs; or,

(ii) ***identity except a stochastic i-th row***, when there is no anchor in the virtual convex hull, i.e., $\Theta_k^i \cap \kappa = \emptyset$; or,

(iii) ***identity except a strictly sub-stochastic i-th row***, when there is at least one anchor in the virtual convex hull[6].

In the next section, we provide sufficient conditions under which the iterative localization algorithm, Eq. (5.13), tracks the true locations of the agents.

[6]Note that if an agent lies inside a virtual convex hull of three anchors, then practically by assigning a zero weight on its past, the agent can find its exact location and take the role of an anchor, which could subsequently increase the convergence rate of the algorithm.

5.4 Convergence analysis

We start this section by reviewing the main results on the convergence of (sub-) stochastic LTV systems that was provided in Chapter 2. We will then adapt these results to investigate the convergence of our localization algorithm represented by Eq. (5.13). Recall the following Assumptions from Section 2.4 on the update at time k:

A0: When the updating agent, i, has no neighboring anchor, we have

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} = 1, \tag{5.19}$$

resulting in a stochastic $\mathbf{P}_k$.

A1: When the updating agent, i, has no anchor but at least one agent as a neighbor, the weights are such that

$$0 < \beta_1 \leq (\mathbf{P}_k)_{i,l} < 1, \qquad \forall l \in \mathcal{D}_k^i,\ \beta_1 \in \mathbb{R}. \tag{5.20}$$

A2: When the updating agent updates with an anchor, we have

$$\sum_{l \in \mathcal{D}_k^i \cap \Omega} (\mathbf{P}_k)_{i,l} \leq \beta_2 < 1 \tag{5.21}$$

resulting in a sub-stochastic $\mathbf{P}_k$.

By defining a slice, $\mathbf{M}_j$, as the smallest product of consecutive system matrices, such that: (i) the infinity norm of each slice is less than one; *and*, (ii) the

entire sequence of system matrices is covered by non-overlapping slices, the following theorem from Section 2.4 characterizes the asymptotic behavior of (sub-) stochastic LTV systems under the above Assumptions:

Theorem 1. *With Assumptions **A0-A2**, the following LTV system*

$$\mathbf{x}_{k+1} = \mathbf{P}_k \mathbf{x}_k, \tag{5.22}$$

converges to zero if any one of the following holds:

(i) *Each slices has a bounded length, i.e.*

$$|\mathbf{M}_j| \leq N < \infty, \forall j,\ N \in \mathbb{N}; \tag{5.23}$$

(ii) *There exists a set, $\boldsymbol{J}_1$, consisting of an infinite number of slices such that*

$$|\mathbf{M}_j| \leq N_1 < \infty, \forall \mathbf{M}_j \in \mathbf{J}_1, \text{ and } |\mathbf{M}_j| < \infty, \forall \mathbf{M}_j \notin \mathbf{J}_1; \tag{5.24}$$

(iii) *For every $i \in \mathbb{N}$, there exists a set, $\boldsymbol{J}_2$, of slices such that*

$$\exists \mathbf{M}_j \in J_2 : \quad |\mathbf{M}_j| \leq \frac{1}{\ln(\beta_1)} \ln\left(\frac{1 - e^{(-\gamma_2 i^{-\gamma_1})}}{1 - \beta_2}\right) + 1, \tag{5.25}$$

for some $\gamma_1 \in [0, 1]$, $\gamma_2 > 0$, and $|\boldsymbol{M}_j| < \infty, j \notin J_2$.

We now adapt the above LTV results to the distributed localization setup described in Section 5.1. Recall that the i-th row of the system matrix, $\mathbf{P}_k$ in Section 5.3, collects the agent-to-agent barycentric coordinates corresponding

to agent i and its neighbors. We now relate the slice representation in Theorem 1 to the information flow in the network: Each slice, $\mathbf{M}_j$, is initiated with a strictly sub-stochastic update, i.e., when one agent with unknown location *directly* receives information from an anchor by having this anchor in its virtual convex hull. On the other hand, a slice, $\mathbf{M}_j$, is terminated after the information from the anchor(s) is propagated through the network and reaches every agent either directly or indirectly. Here, *directly* means that an agent has an anchor in its virtual convex hull; while *indirectly* means that an agent has a neighbor in its virtual convex hull, which has previously received the information (either directly or indirectly) from an anchor. Once the anchor information reaches every agent in the network, the slice notion and Theorem 1 provide the conditions on the rate, at which this information should propagate for convergence. We proceed with the following lemma.

Lemma 12. *Under the conditions **B0-B4** and no noise, the product of system matrices, $\mathbf{P}_k$'s, in the LTV system, Eq.* (5.13)*, converges to zero if one of the conditions in Theorem 1 holds.*

Proof. We need to show that the Assumptions **A0-A3** follow from **B0-B4**. First note that Assumption **A0** is followed from the fact that barycentric coordinates always sum to one. Also note that Eqs. (5.12) and (5.18) result in Assumption **A1** if we set $\beta_1 = \min\{\beta, \alpha'\}$. Next, we note that **A3** is implied

by **B**3. This is because if $\Theta_k^i \cap \kappa$ is not empty, we can write

$$\sum_{j \in \Theta_k^i \cap \Omega} (\mathbf{P}_k)_{i,j} = 1 - \sum_{m \in \Theta_k^i \cap \kappa} (\mathbf{B}_k)_{i,m}, \tag{5.26}$$

where we used the fact that the weighted barycentric coordinates sum to one. When there is only one anchor among the nodes forming the virtual convex hull, and the minimum weight is assigned to this anchor, Eq. (5.17), the right hand side of Eq. (5.26) is maximized. This provides an upper bound on the i-th row sum:

$$\sum_{j \in \Theta_k^i \cap \Omega} (\mathbf{P}_k)_{i,j} \leq 1 - \alpha < 1, \tag{5.27}$$

which ensures **A3** with $\beta_2 = 1 - \alpha$. In addition, Lemma 11 ensures that each agent updates infinitely often with different neighbors. Subsequently, each slice is completed after all agents receive anchor information (at least once) either directly or indirectly, and the asymptotic convergence of Eq. (5.13) follows under the conditions in Theorem 1. □

The following theorem completes the localization algorithm for mobile multi-agent networks in $\mathbb{R}^2$.

Theorem 5. *Consider a network of M (possibly mobile) anchors and N mobile agents moving in a finite and bounded region in $\mathbb{R}^2$. Then, under Assumptions **B0-B4** and no noise, for any (random or deterministic) motion*

that satisfies one of the conditions in Theorem 1, the solution of Eq. (5.13) *asymptotically converges to the true agent locations.*

Proof. In order to show the convergence to the true locations, we show that the error between the location estimate, $\mathbf{x}_k$, and the true location, $\mathbf{x}_k^*$, goes to zero. To find the error dynamics, note that the true agent locations follow:

$$\mathbf{x}_{k+1}^* = \mathbf{P}_k \mathbf{x}_k^* + \mathbf{B}_k \mathbf{u}_k + \widetilde{\mathbf{x}}_{k+1}. \tag{5.28}$$

Subtracting Eq. (5.13) from Eq. (5.28), we get the network error

$$\mathbf{e}_{k+1} \triangleq \mathbf{x}_{k+1}^* - \mathbf{x}_{k+1} = \mathbf{P}_k(\mathbf{x}_k^* - \mathbf{x}_k) = \mathbf{P}_k \mathbf{e}_k, \tag{5.29}$$

which goes to zero when

$$\lim_{k\to\infty} \prod_{l=0}^{k} \mathbf{P}_l = \mathbf{0}_{N\times N}, \tag{5.30}$$

from Lemma 12. □

The following theorem characterizes the number of anchors that are required for the iterative localization algorithm, Eq. (5.13), to converge to the true agent locations.

Theorem 6. *Under Assumptions* ***B0-B4*** *and no noise, Eq.* (5.13) *tracks the true agent locations in* $\mathbb{R}^2$*, when the number of agents,* N*, and the number of anchors,* M*, follow:*

$$M \;\geq\; 1, \tag{5.31}$$

$$N + M \geq 4. \tag{5.32}$$

Proof. Let us first consider the requirement of at least one anchor. Without an anchor, a strictly sub-stochastic row never appears in $\mathbf{P}_k$, making $\mathbf{P}_k$ stochastic at each time and hence the infinite product of $\mathbf{P}_k$'s is also stochastic and not zero. With at least one anchor, strictly sub-stochastic rows, following Eq. (5.27), appear in $\mathbf{P}_k$'s, and zero convergence of the error dynamics, Eq. (5.29), follows. Next, exactly 3 nodes (agents and/or anchors) are required to form a (virtual) convex hull in $\mathbb{R}^2$. Thus, when the total number of nodes, $N + M$, is 3 or less, no agent can find 3 other nodes to find (virtual) convex hulls. Using the same argument as the one in the proof of Lemma 11, we can show that with at least one anchor and at least 4 total nodes, any agent with unknown location infinitely finds itself in arbitrary (virtual) convex hulls. Thus, Theorem 1 is applicable and the proof follows. □

5.5 Localization under imperfect measurements

The noise on distance measurements and motion degrades the performance of the localization algorithm, as expected, and in certain cases the location error is as large as the region of motion; this is shown experimentally in Sec-

tion 5.6. In what follows, we discuss the modifications, **M1-M3**, to the proposed algorithm to address the noise in case of motion, $\widehat{\mathbf{x}}_k^i$, and on the distance measurements, $\widehat{d}_k^{ij}$.

Discard unreliable Cayley-Menger determinants

To get meaningful values for the areas and volumes in $\mathbb{R}^2$ according to Eq. (B.3), the corresponding Cayley-Menger determinants computed with perfect distance measurements must be negative. Therefore, we suggest the first modification to the algorithm as follows:

M1: An agent does not perform an inclusion test if the corresponding Cayley-Menger determinant is positive.

Inclusion test error

Even if the inclusion test results may not be accurate due to the noise on the motion, which corresponds to imperfect location updates *at each and every iteration*. To tackle this issue, we propose the following modification to the algorithm:

M2: If the inclusion test is passed at time k by a triangulation set, Θ_k^i, agent i performs an update only if

$$\epsilon_k^i = \left|\frac{\sum_{j\in\Theta_k^i} A_{\Theta_k^i\cup\{i\}\setminus j} - A_{\Theta_k^i}}{A_{\Theta_k^i}}\right| < \epsilon, \tag{5.33}$$

in which $\epsilon_k^i \geq 0$ is the *inclusion test relative error* for agent i at time k, and $\epsilon \geq 0$ is a design parameter.

Convexity

Finally, recall that if the inclusion test is passed at time k, by a triangulation set, Θ_k^i, the updating agent, i, updates its location estimate, $\mathbf{x}_k^{i*}$, as a convex combination of the location estimates of the nodes in Θ_k^i. In order to guarantee the convexity in the updates in presence of noise, we consider the following modification to the algorithm:

M3: If the inclusion test is passed at time k by a triangulation set, Θ_k^i, and Eq. (5.33) holds at time k, agent i (randomly) chooses one of the i-visited nodes from the triangulation set Θ_k^i, say agent j, and finds the corresponding weight, a_k^{ij}, as follows:

$$a_k^{ij} = 1 - a_k^{im} - a_k^{in}, \qquad \{j, m, n\} \in \Theta_k^i, \tag{5.34}$$

assuming that $a_k^{im} + a_k^{in} < 1$.

5.6 Simulations

In this section, we first provide simulation results in noiseless scenarios and then examine the effects of noise.

Localization without noise

We now consider the noiseless scenarios, i.e., we assume that $n_k^i = 0$ and $r_k^{ij} = 0$, in Eqs. (5.2) and (5.3). In the beginning, all nodes (agents and anchor(s)) are randomly deployed within the region of $x \in [-5\ 15],\ y \in [-5\ 15]$ in $\mathbb{R}^2$. For the simulations, we consider Random Waypoint mobility model, [139], for the motion in the network, which has been the predominant motion model in the localization literature over the past decade, see e.g., [35–37, 161, 162]. We note that better performance may be achieved if the agents follow some deterministic (mission-related) motion model. However, our localization algorithm converges if the motion is such that the slice lengths grow slower than the exponential rate in Eq. (5.25). For all mobile nodes, $i \in \Theta$, we chose $d_{k\to k+1}^i$ and $\theta_{k\to k+1}^i$ in Eq. (5.6) as random variables with uniform distributions over the intervals of $[0\ 5]$ and $[0\ 2\pi]$, respectively. Each agent can only communicate with the nodes within its communication radius, which is set to $r = 2$ for all simulations. For each simulation, we assume exactly one fixed anchor, i.e., $M = 1$. In all simulations, we set $\alpha_k = 0.2$ to ensure that the agents do not completely forget the past information and $\alpha = \alpha' = 0.1$ to guarantee an adequate contribution from the anchor(s) and agents.

Fig. 5.7 (Left) shows the random trajectories of $N = 3$ mobile agents and $M = 1$ anchor for the first 25 iterations, where red triangle indicates

anchor and filled circles show the initial locations of the agents. To characterize

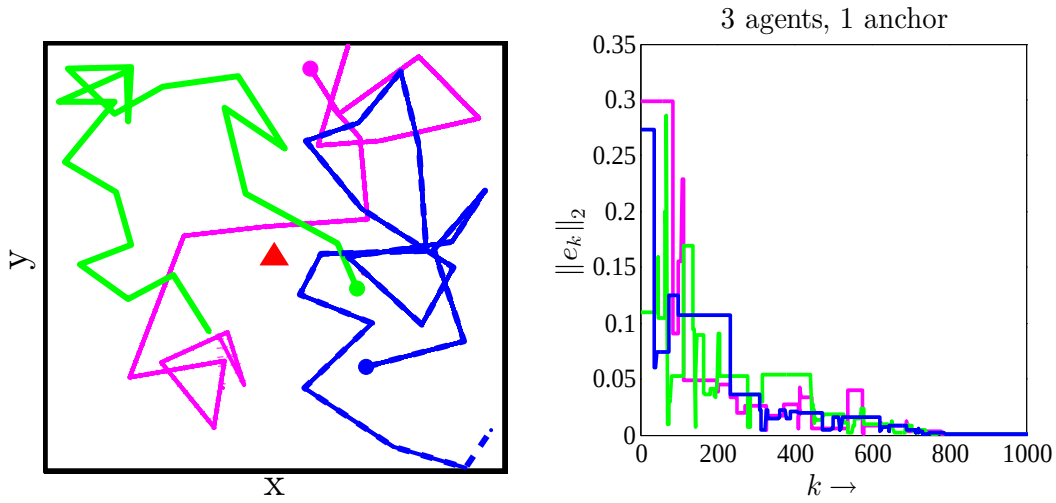

Figure 5.7: A network of 3 mobile agents and 1 anchor: (Left) Motion model. (Right) Convergence.

the convergence, we choose the 2-norm of the error vector normalized with respect to the dimensions of the region as follows:

$$\|\mathbf{e}_k\|_2 = \frac{1}{2}\sqrt{\left(\frac{\mathbf{x}_k - \mathbf{x}_k^*}{\Delta x}\right)^2 + \left(\frac{\mathbf{y}_k - \mathbf{y}_k^*}{\Delta y}\right)^2}, \tag{5.35}$$

in which $\Delta x = 20$ and $\Delta y = 20$ denote the lengths of the region, which is a square in this case. The division by Δx and Δy is done to obtain a normalized error. It can be verified that the maximum error with this normalization is $\frac{1}{\sqrt{2}}$. The algorithm converges when $\|\mathbf{e}_k\|_2 \to 0$. As shown in Fig. 5.7 (Right), the localization algorithm, Eq. (5.13), tracks the true agent locations. The simulation results for a network of 10 and 100 mobile agents and one mobile anchor is illustrated in Fig. 5.8 (Left) and (Right), respectively.

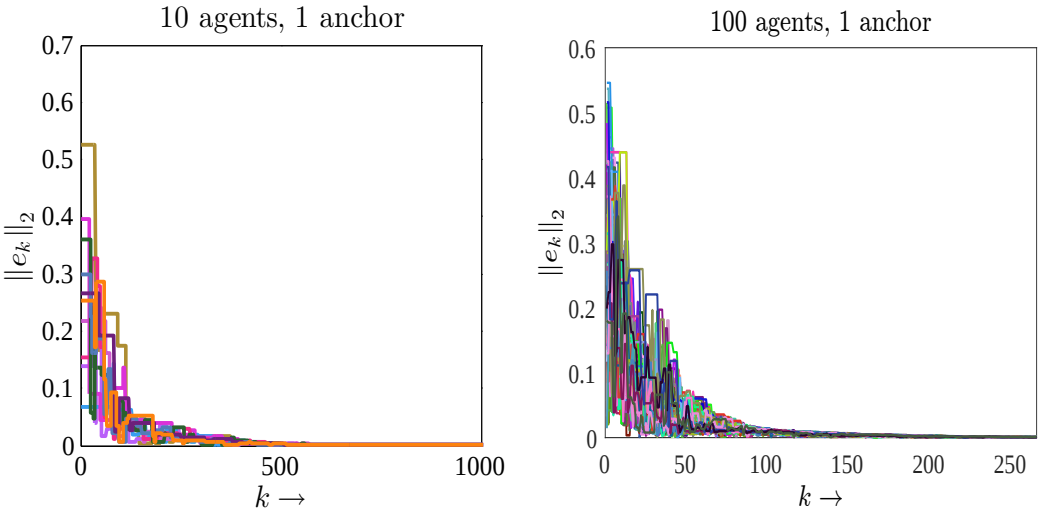

Figure 5.8: Convergence of a network with one anchor: (Left) 10 mobile agents (Right) 100 mobile agents.

Before we examine the effects of noise on the proposed localization algorithm, let us study the case where there is no anchor in the network, i.e., Eq (5.31) is not satisfied; In most localization algorithms, $m+1$ anchors are required to localize an agent with unknown location in $\mathbb{R}^m$ without ambiguity. However, when the nodes are mobile, *exactly one anchor* is sufficient to inject valuable information and the motion of the agents provides the remaining degrees of freedom. When there is no anchor in the network, all updates are stochastic, and we get

$$\mathbf{e}_{k+1} = \mathbf{P}_k \mathbf{e}_k, \quad \text{with} \quad \rho(\mathbf{P}_k) = 1, \forall k,$$

where $\rho(.)$ denotes the spectral radius, i.e., a neutrally stable system, which leads to a bounded steady-state error in location estimates. In other words,

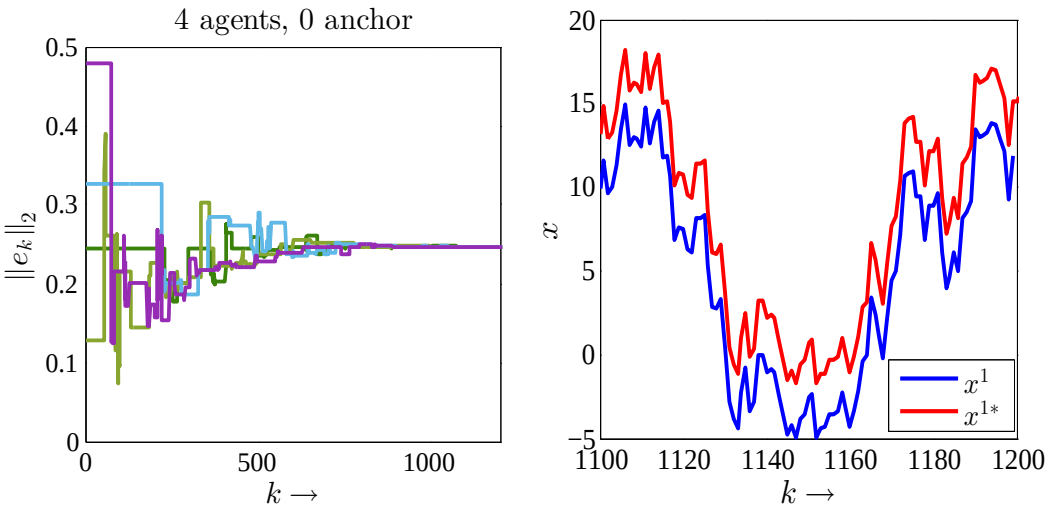

Figure 5.9: Steady state error for a network of 4 mobile agents and no anchor.

relative locations are tracked because the motion removes certain ambiguity from the locations. However, absolute tracking without any ambiguity requires at least one anchor. This is illustrated in Fig. 5.9 for a network of $M = 4$ agents and no anchor. Fig. 5.9 (Left) shows the 2-norm of error, and red and blue curves in Fig. 5.9 (Right) represent the exact and estimated values of x coordinate, for the last 100 iterations of a simulation.

Localization with noise

To examine the effects of noise on the proposed localization algorithm, we let the noise on the motion of agent $i \in \Psi$ at time k, n_k^i to be $\pm 1\%$ of the magnitude of the motion vector, $\widetilde{\mathbf{x}}_k^i$, and the noise on distance measurements, r_k^{ij} to be $\pm 10\%$ of the actual distance between nodes i and j, d_k^{ij}. As shown in

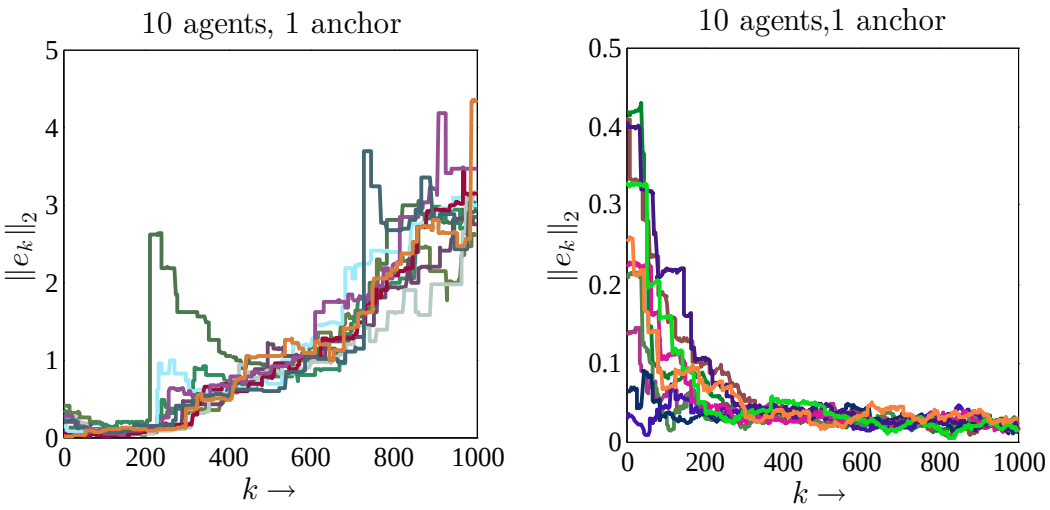

Figure 5.10: Effect of noise on the convergence: (Left) Original algorithm. (Right) Modified algorithm.

Fig. 5.10 (Left) this amount of noise leads to an unbounded error, which is due to incorrect inclusion test results and the continuous location drifts because of the noise on the distance measurements and the noise on motion, respectively. However, by modifying the algorithm according to Section 5.5, it can be seen in Fig. 5.10 (Right) that the normalized localization error is reduced to less than 5% for one simulation. Finally, in Fig. 5.11 we provide 20 Monte Carlo simulations for a network of $M = 1$ mobile anchor and $N = 10$ and $N = 20$ mobile agents in presence of noise. In this simulations we add $\pm 10\%$ noise on distance measurements, and $\pm 1\%$ noise on the motion.

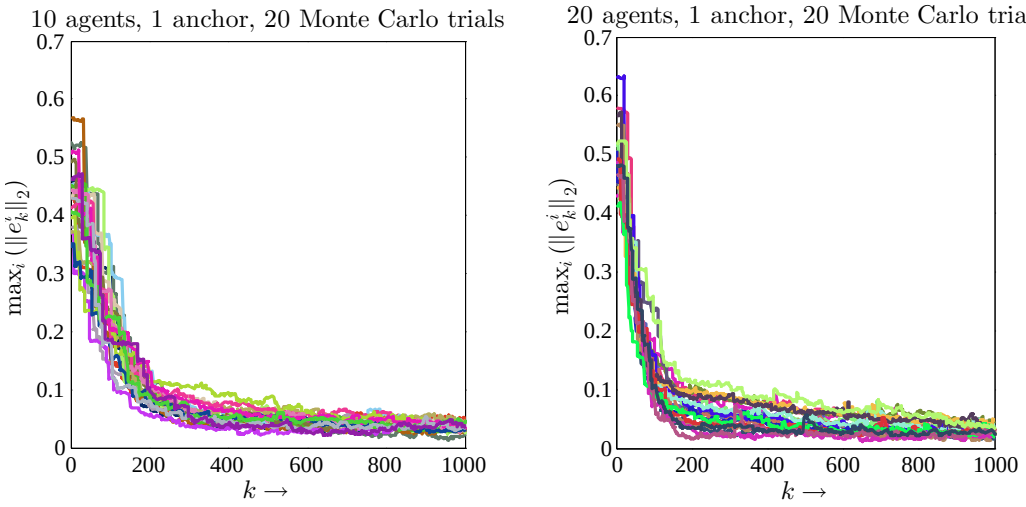

Figure 5.11: 20 Monte Carlo trials with noise: (Left) $N = 10,\ M = 1$. (Right) $N = 20,\ M = 1$.

Performance evaluation

We now evaluate the performance of our algorithm in contrast with some well-known localization methods; MCL [36], MSL* [35], Dual MCL [40], and Range-based SMCL [37]. In Fig. 5.12, we compare the localization error in the Virtual Convex Hull (VCH) algorithm with MCL, MSL*, and Range-based SMC. As shown in Table 5.1, in these algorithms node density, n_d, and anchor (seed) density, n_s, denote as the average number of nodes and anchors in the neighborhood of an agent, respectively. Total number of agents and anchors can therefore be determined by knowing these densities as well as the area of the region. We consider $N = 20$ agents and $M = 2$ anchors, and use the same metric as used in [35–37], i.e., the location error as a percentage of the

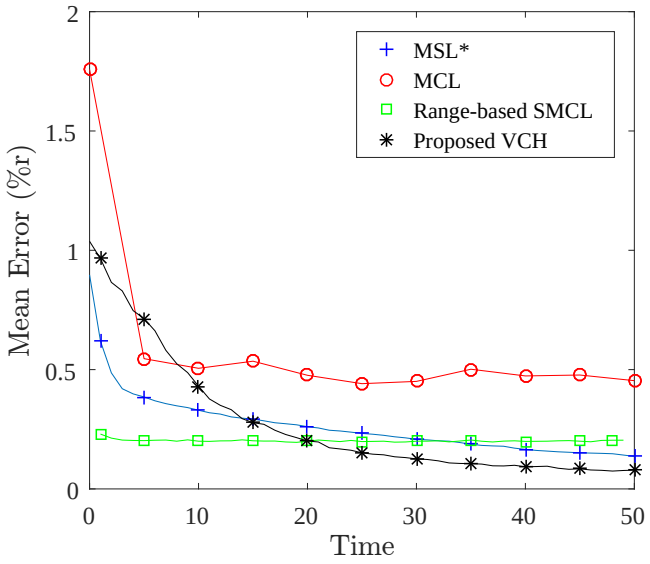

Figure 5.12: Accuracy comparison.

communication range. Each data point in Fig. 5.12 is computed by averaging the results of 20 simulation experiments. We keep the other parameters the same as described earlier in this section. With high measurement noise, i.e., in the presence of 10% noise on the range measurements and 1% noise on the motion, our algorithm outperforms MCL, MSL* and Range-based SMCL after 20 iterations. Clearly, the localization error in our algorithm decreases as the amount of noise decreases, and our algorithm converges to the exact agent locations in the absence of noise. Table 5.1 summarizes the performance of the proposed VCH algorithm in comparison to the above methods.

Localization algorithm	**Simulation environment, Network size, Communication range**	**Average error %r**
MCL	500*500, n_d=10, n_s=1, r =50	0.2 - 0.6
Dual MCL	250*250, n_d=10, n_s=2, r =25	0.4 - 0.6
MSL*	500*500, n_d=10, n_s=2, r =100	0.1 - 0.5
Range-based SMCL	1*1, n_d=13.9, n_s=1.3, r =0.125	0.2 - 0.4
Proposed VCH	20*20, N=20, M=2, r=2	< 0.1

Table 5.1: Comparative performance of localization algorithms.

5.7 Summary

In this chapter, we provide a distributed algorithm to localize a network of mobile agents moving in a bounded region of interest in $\mathbb{R}^2$. Assuming that each agent knows a noisy version of its motion and the distances to the nodes in its communication radius, we provide a geometric framework, which allows an agent to keep track of the distances to any previously visited nodes, and find the distance between a neighbor and any virtual location where it has exchanged information with other nodes in the past. Since agents are mobile, they may not be able to find neighbors to perform distributed updates at any time. To avoid this issue, we introduce the notion of a virtual convex hull, which forms the basis of a range-based localization algorithm in mobile networks. We abstract the algorithm as an LTV system with (sub-) stochastic

matrices, and show that it converges to the true locations of agents under some mild regularity conditions on update weights. We further show that exactly one anchor suffices to localize an arbitrary number of mobile agents when each agent is able to find appropriate (virtual) convex hulls. We evaluate the performance of the algorithm in presence of noise and provide modifications to the proposed algorithm to address noise on motion and on distance measurements.

Appendix A

An overview of Bayesian approaches for localization in mobile networks

In what follows, we discuss the basic principles and characteristics of predominant probabilistic approaches for distributed localization in mobile networks that can be broadly characterized as Bayesian approaches.

Problem Formulation: Consider a network of N mobile agents collected in a set Ω and M anchors in a set κ in $\mathbb{R}^m$, $m \geq 1$. The N agents in Ω have unknown locations, while the M anchors in κ have known locations. Let $\mathbf{x}_k^i \in \mathbb{R}^m$ be a row vector that denotes the location of a (possibly mobile)

node i at time k, where $k \geq 0$ is the discrete-time index. We describe the evolution of the agent locations as follows:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{v}_k), \tag{A.1}$$

in which $\mathbf{x}_k \in \mathbb{R}^{N \times m}$ is the concatenated state of all agents in the network at time k, the function, f, possibly non-linear, captures the temporal evolution of the locations, and $\mathbf{v}_k$ represents the uncertainty in the location evolution at time k. Similarly, assume $\mathbf{z}_k^i$ to denote the local measurement at agent i and time k. Let $\mathbf{z}_k$ be the collection of all measurements at time k leading to the following global measurement model:

$$\mathbf{z}_k = g(\mathbf{x}_k, \mathbf{n}_k), \tag{A.2}$$

in which $\mathbf{n}_k$ captures the measurement noise. The function, g, denotes the measurement technology. We denote the set of all measurements acquired at time steps, $1, \ldots, k$, by the set

$$\mathbf{Z}_k = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}.$$

The localization problem is to estimate the locations, $\mathbf{x}_k^i$, of the mobile agents in the set Ω given the measurements in the set $\mathbf{Z}_k$.

Baysian Estimation: In general, Bayesian filtering refers to the process of using Bayes' rule to estimate the parameters of a time-varying system,

which is indirectly observed through some noisy measurements. In the context of localization, given the history of the measurements up to time k, $\mathbf{Z}_k$ as defined in Eq. (A.1), Bayesian filtering aims to compute the posterior density, $p(\mathbf{x}_k \mid \mathbf{Z}_k)$, of the agents' locations, $\mathbf{x}_k$, i.e., the agent's *belief* about their location at time k. The goal of localization is to make the belief for each agent as close as possible to the actual distribution of the agent's location, which has a single peak at the true location and is zero elsewhere. Before we explain the Baysian estimation process, let us define the following terms: The *dynamic model*, $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, captures the dynamics of the system, and corresponds to the motion model in the context of localization; it describes the agents' locations at time k, given that they were previously located at $\mathbf{x}_{k-1}$. The *measurement model*, $p(\mathbf{z}_k \mid \mathbf{x}_k)$, represents the distribution of the measurements given the agents' locations at time k. The measurement model captures the error characteristics of the sensors, and describes the likelihood of taking measurement, $\mathbf{z}_k$, given that the agents are located at $\mathbf{x}_k$. The Baysian estimation process can then be summarized in the following steps:

1. **Initialization**: Before the agents start acting (moving) in the environment, they may have initial beliefs about where they are. The process then starts with a prior distribution of agent locations, $p(\mathbf{x}_0)$, which represents the information available on the initial locations of the agents

before taking any measurements. For example, in robotic networks such information may be available by providing the robots with the map of the environment. When prior information on agent locations is not available, the prior distribution can be assumed to be uniform.

2. **Prediction**: Suppose the motion model and the belief at time $k-1$, $p(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1})$, are available. Then the predictive distribution of the agents' locations at time k can be computed as follows:

$$p(\mathbf{x}_k \mid \mathbf{Z}_{k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}. \tag{A.3}$$

3. **Update**: Whenever new measurements, $\mathbf{z}_k$, is available, the agents incorporate the measurements into their beliefs to form new beliefs about their locations, i.e., the predicted estimate gets updated as follows:

$$p(\mathbf{x}_k \mid \mathbf{Z}_k) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{Z}_{k-1})}{p(\mathbf{z}_k \mid \mathbf{Z}_{k-1})}, \tag{A.4}$$

in which the normalization constant,

$$p(\mathbf{z}_k \mid \mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{Z}_{k-1}) d\mathbf{x}_k, \tag{A.5}$$

depends on the measurement model and guarantees that the posterior over the entire state space sums up to one.

In this process, it is standard to assume that the current locations of the mobile agents, $\mathbf{x}_k$, follow the *Markov* assumption, which states that the agent

locations at time k depends only on the previous location, $\mathbf{x}_{k-1}$, i.e.,

$$p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{Z}_{k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}). \tag{A.6}$$

In other words, according to the Markovian dynamic model, the current locations contain all relevant information from the past. Otherwise, as the process continues and the number of sensor measurements increases, the complexity of computing the posterior distributions grows exponentially over time. Under the Markov assumption, the computation cost and memory demand decrease and the posterior distributions can be efficiently computed without losing any information, making the localization process usable in real-time scenarios.

Bayes filters provide a probabilistic framework for recursive estimation of the agents' locations. However, their implementation requires specifying the measurement model, the dynamic model, and the the posterior distributions. The properties of the different implementations of Bayes filters strongly differ in how they represent probability densities over the state, $\mathbf{x}_k$, [163]. The reader is referred to [163] for a complete survey of the Bayesian filtering techniques. In what follows, we briefly review Kalman filters and particle filters that are the most commonly used variants of Bayesian methods.

Kalman Filtering: When both the dynamic model and the measurement model can be described using Gaussian density functions, and the initial distribution of the agent locations is also Gaussian, it is possible to use Kalman

filters, to derive an exact analytical expression to compute the posterior distribution of the agent locations. Mathematically, Kalman filters can be used if the dynamic and measurement models can be expressed as follows:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_k \mathbf{x}_k + \mathbf{v}_k, \\ \mathbf{z}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{n}_k, \end{aligned} \tag{A.7}$$

in which the process noise,

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k),$$

and the measurement noise,

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k),$$

are Gaussian with zero mean and covariance matrices of $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively, and prior distribution is also Gaussian:

$$\mathbf{x}_0 \sim \mathcal{N}(\mu_0, \mathbf{\Sigma}_0).$$

In Eq. (A.7), the matrix $\mathbf{F}_k$ is the transition matrix and $\mathbf{H}_k$ is the measurement matrix.

Due to linear Gaussian model assumptions in Eq. (A.7), the posterior distribution of agent locations is also Gaussian, and can be exactly computed by implementing the prediction, Eq. (A.3), and update, Eq. (A.4), steps using efficient matrix operations and without any numerical approximations. The

Kalman filter algorithm can be represented with the following recursive algorithm, [164]:

$$\begin{aligned} p(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}) &= \mathcal{N}(\mathbf{m}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}), \\ p(\mathbf{x}_k \mid \mathbf{Z}_{k-1}) &= \mathcal{N}(\mathbf{m}_{k|k-1}, \mathbf{P}_{k|k-1}), \\ p(\mathbf{x}_k \mid \mathbf{Z}_k) &= \mathcal{N}(\mathbf{m}_{k|k}, \mathbf{P}_{k|k}), \end{aligned}$$

such that

$$\begin{aligned} \mathbf{m}_{k|k-1} &= \mathbf{F}_k \mathbf{m}_{k-1|k-1}, \\ \mathbf{P}_{k|k-1} &= \mathbf{Q}_{k-1} + \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T, \\ \mathbf{m}_{k|k} &= \mathbf{m}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \mathbf{m}_{k|k-1}), \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}, \end{aligned}$$

where

$$\begin{aligned} \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k, \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}, \end{aligned}$$

are the covariance of $\mathbf{z}_k - \mathbf{H}_k \mathbf{m}_{k|k-1}$, and the Kalman gain, respectively.

Kalman filters are optimal estimators when the model is linear and Gaussian. They provide efficient, accurate results if the uncertainty in the location estimates is not too high, i.e., when accurate sensors with high update rates are used, [163]. However, in many situations such assumptions do not hold,

and no analytical solution can be provided. In such scenarios, the extended Kalman filter can be used that approximates the non-linear and non-Gaussian dynamic and measurement models by linearizing the system using first-order Taylor series expansions.

Particle Filters: As discussed earlier, extended Kalman filter results in a Gaussian approximation to the posterior distribution of the agent locations, and generates large errors if the true distribution of the belief is not Gaussian. In such cases, the *Particle filters* that generalize the traditional Kalman filtering methods to non-linear, non-Gaussian systems can provide more accurate results. Particle filters, also known as sequential Monte Carlo methods, provide an effective framework to track a variable of interest as it evolves over time, when the underlying system is non-Gaussian, non-linear, or multidimensional. Unlike other Bayesian filters, sequential Monte Carlo method is very flexible, easy to implement and suitable for parallel processing, [161]. Since the dynamic and measurement models are often non-linear and non-Gaussian, sequential Monte Carlo methods are in particular widely employed to solve the localization problem in mobile networks. The particle filter implementation for the localization problem in mobile wireless sensor networks is a recursive Bayesian filter that constructs a sample-based representation of the entire probability density function, and estimates the posterior distribution of agents' locations conditioned on their observations. The key idea is to represent the

required posterior probability density function by a set of random samples or *particles*, i.e., candidate representations of agents' coordinates, weighted according to their likelihood and to compute estimates based on these samples and weights. At time k, the location estimate of the i-th mobile agent, is represented as a set of N_s samples (particles) , i.e.,

$$S_k^i = \{\mathbf{x}_k^j, w_k^j\},\ j = 1, \ldots, N_s, \tag{A.8}$$

in which j indicates the particle index, and the weight, w_k^j, also referred to as *importance factor*, defines the contribution of the j-th particle to the overall estimate of agent i's location. These samples form a discrete representation of the probability density function of the i-th moving agent, and the importance factor is determined by the likelihood of a sample given agent i-th latest observation. The sequential Monte Carlo localization process for each agent can then be summarized in the following steps:

1. **Initialization**: At this stage, N_s samples are randomly selected from the prior distribution of agent locations, $p(\mathbf{x}_0)$.

2. **Prediction**: In this step, the effect of the action, from time $k-1$ to k, e.g., the movement of the agent according to the dynamic model, is taken into account, and N_s new samples are generated to represent the current location estimate. At this point, a random noise is also added to the particles in order to simulate the effect of noise on location estimates.

3. **Update**: At this stage, sensor measurements are introduced to correct the outcome of the prediction step; each particle's weight is re-evaluated based on the latest sensory information available, in order to accurately describe the moving agents' probability density functions.

4. **Resampling**: At this step the particles with very small weights are eliminated, and get replaced with new randomly generated particles so that the number of particles remain constant.

By iteratively applying the above steps, the particle population eventually converges to the true distribution. Sequential Monte Carlo methods have several key advantages compared with the previous approaches: (i) due to their simplicity, they are is easy to implement; (ii) they are able to represent noise and multi-modal distributions; and (iii) since the posterior distribution can be recursively computed, it is not required to keep track of the complete history of the estimates, which in turn reduces the amount of memory required, and can integrate observations with higher frequency. However, in general sequential Monte Carl methods are very time-consuming because they need to keep sampling and filtering until enough samples are obtained for representing the posterior distribution of a moving node's position, [133]. Moreover, they often require a high density of anchors to achieve accurate location estimates.

Appendix B

Localization in dynamic networks

B.1 Convex hull inclusion test

In any arbitrary dimension, m, a convex hull inclusion test can be described as follows, [95]:

$$i \in \mathcal{C}(\mathcal{N}_k^i), \qquad \text{if } \sum_{j \in \mathcal{N}_k^i} A_{\mathcal{N}_k^i \cup \{i\} \setminus j} = A_{\mathcal{N}_k^i}, \tag{B.1}$$

$$i \notin \mathcal{C}(\mathcal{N}_k^i), \qquad \text{if } \sum_{j \in \mathcal{N}_k^i} A_{\mathcal{N}_k^i \cup \{i\} \setminus j} > A_{\mathcal{N}_k^i}, \tag{B.2}$$

in which $\mathcal{N}_k^i$ denotes the neighbors of node i at time k, $\mathcal{C}(\cdot)$ is the convex hull, and $A_{\mathcal{N}_k^i}$ represents the generalized volume of $\mathcal{C}(\mathcal{N}_k^i)$, and can be computed by using the Cayley-Menger determinant, [144]. Note that, " $\setminus$ " in above

equations denotes the set difference, thus $A_{\mathcal{N}_k^i \cup \{i\} \setminus j}$ is the generalized volume of the set $\mathcal{N}_k^i$ with node i added and node j removed. Fig. B.1, depicts the

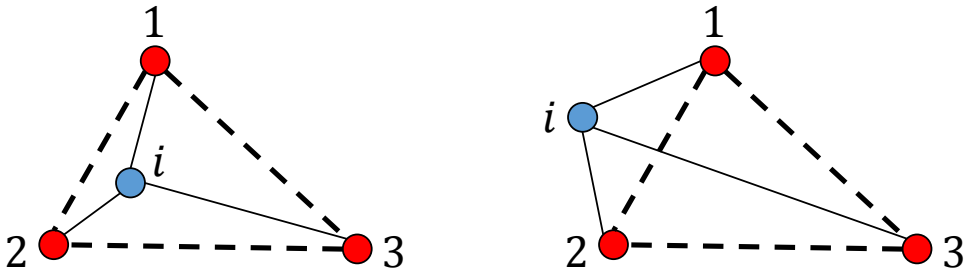

Figure B.1: Inclusion test in $\mathbb{R}^2$.

inclusion test in $\mathbb{R}^2$. In Fig. B.1 (Left) node i lies inside the triangle formed by the neighboring nodes, 1, 2 and 3, whereas in Fig. B.1 (Right) the inclusion test is not passed.

B.2 Cayley-Menger determinant

Consider a set of $m+1$ points, Θ_ℓ, in m-dimensional Euclidean space. Cayley-Menger (CM) determinant is the determinant of an $(m+2)\times(m+2)$ symmetric matrix, which uses the pairwise distance information in the set Θ_ℓ, to compute the hypervolume, A_{Θ_ℓ}, of their convex hull, $\mathcal{C}(\Theta_\ell)$. The CM determinant is given by

$$A^2_{\Theta_\ell} = \frac{1}{s_{m+1}} \begin{vmatrix} 0 & \mathbf{1}^T_{m+1} \\ \mathbf{1}_{m+1} & \mathbf{D} \end{vmatrix}, \tag{B.3}$$

in which $\mathbf{1}_{m+1}$ denotes an $m+1$-dimensional column vector of 1s, $\mathbf{D} = \{d_{\ell j}^2\},\ \ell, j \in \Theta_l$, is the $(m+1) \times (m+1)$ matrix of squared distances, $d_{\ell j}$, within the set, Θ_ℓ, and

$$s_m = \frac{2^m (m!)^2}{(-1)^{m+1}}, \qquad m \in \{0, 1, 2, \ldots\}. \tag{B.4}$$

The second and third coefficients in the above sequence that are relevant in $\mathbb{R}^2$ and $\mathbb{R}^3$ are -16 and 288, respectively. Cayley-Menger determinant provides a simple equation that extends to any dimensions. It computes the hypervolume of m-dimensional simplexes and reduces to, e.g., Herons formula for the area of a triangle in $\mathbb{R}^2$, and volume of a tetrahedron in $\mathbb{R}^3$.

Bibliography

[1] H. H. Rosenbrook. "The Stability of Linear Time-dependent Control Systems". In: *International Journal of Electronics and Control* 15.1 (1963), pp. 73–80.

[2] R. K. Brayton and C. Tong. "Stability of dynamical systems: A constructive approach". In: *IEEE Transactions on Circuits and Systems* 26.4 (1979), pp. 224–234.

[3] A. Ilchmann, D. H. Owens, and D. Pratzel-Wolters. "Sufficient conditions for stability of linear time-varying systems". In: *Systems and control letters* 9.2 (1987), pp. 157–163.

[4] K. S. Tsakalis and P. A. Ioannou. *Linear time-varying systems: control and adaptation*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1993.

[5] J. J. DaCunha. "Stability for time varying linear dynamic systems on time scales". In: *Journal of Computational and Applied Mathematics* 176.2 (2005), pp. 381 –410.

[6] V. N. Phat and P. Niamsup. "Stability of linear time-varying delay systems and applications to control problems". In: *Journal of Computational and Applied Mathematics* 194.2 (2006), pp. 343 –356.

[7] M. Wu. "A note on stability of linear time-varying systems". In: *IEEE Transactions on Automatic Control* 19.2 (1974), pp. 162–162.

[8] V. D. Blondel and J. N. Tsitsiklis. "The boundedness of all products of a pair of matrices is undecidable". In: *Systems and Control Letters* 41.2 (2000), pp. 135–140.

[9] D. Estrin et al. "Instrumenting the world with wireless sensor networks". In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on.* Vol. 4. IEEE. 2001, pp. 2033–2036.

[10] G. J. Pottie and W. J. Kaiser. "Wireless integrated network sensors". In: *Communications of the ACM* 43.5 (2000), pp. 51–58.

[11] D. Estrin et al. "Next century challenges: Scalable coordination in sensor networks". In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking.* ACM. 1999, pp. 263–270.

[12] D. C. Steere et al. "Research challenges in environmental observation and forecasting systems". In: *Proceedings of the 6th annual interna-*

tional conference on Mobile computing and networking. ACM. 2000, pp. 292–299.

[13] K. Witrisal et al. "High-Accuracy Localization for Assisted Living: 5G systems will turn multipath channels from foe to friend". In: *IEEE Signal Processing Magazine* 33.2 (2016), pp. 59–70.

[14] L. Militano et al. "Device-to-device communications for 5G internet of things". In: *IOT, EAI* (2015).

[15] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu. "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions". In: *IEEE Communications Magazine* 52.5 (2014), pp. 86–92.

[16] P. Zhang et al. "Cooperative localization in 5G networks: A survey". In: {*ICT*} *Express* 3.1 (2017), pp. 27 –32. ISSN: 2405-9595.

[17] A. Hakkarainen et al. "High-Efficiency Device Localization in 5G Ultra-Dense Networks: Prospects and Enabling Technologies". In: *82nd Vehicular Technology Conference (VTC Fall)*. IEEE. 2015, pp. 1–5.

[18] R. Di Taranto et al. "Location-aware communications for 5G networks: How location information can improve scalability, latency, and robustness of 5G". In: *IEEE Signal Processing Magazine* 31.6 (2014), pp. 102–112.

[19] T. Ojha, S. Misra, and N. S. Raghuwanshi. "Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges". In: *Computers and Electronics in Agriculture* 118 (2015), pp. 66–84.

[20] N. A. Alrajeh, S. Khan, and B. Shams. "Intrusion detection systems in wireless sensor networks: A review". In: *International Journal of Distributed Sensor Networks* (2013).

[21] C. R. Baker et al. "Wireless Sensor Networks for Home Health Care". In: *21st International Conference on Advanced Information Networking and Applications Workshops, AINAW '07.* Vol. 2. 2007, pp. 832–837.

[22] H. Wymeersch, J. Lien, and M. Z. Win. "Cooperative localization in wireless networks". In: *Proceedings of the IEEE* 97.2 (2009), pp. 427–450.

[23] P. Rawat et al. "Wireless sensor networks: A survey on recent developments and potential synergies". In: *The Journal of supercomputing* 68.1 (2014), pp. 1–48.

[24] S. I. Roumeliotis and G. A. Bekey. "Collective localization: A distributed Kalman filter approach to localization of groups of mobile robots". In: *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00.* Vol. 3. IEEE. 2000, pp. 2958–2965.

[25] S. I. Roumeliotis and G. A. Bekey. "Distributed multirobot localization". In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 781–795.

[26] A. Martinelli and R. Siegwart. "Observability analysis for mobile robot localization". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2005, pp. 1471–1476.

[27] A. Martinelli, F. Pont, and R. Siegwart. "Multi-robot localization using relative observations". In: *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE. 2005, pp. 2797–2802.

[28] F. Kong et al. "Mobile robot localization based on extended Kalman filter". In: *The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006.* Vol. 2. IEEE. 2006, pp. 9242–9246.

[29] N. Trawny, S. I. Roumeliotis, and G. B. Giannakis. "Cooperative multi-robot localization under communication constraints". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 4394–4400.

[30] B. S. Cho et al. "A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding". In: *Journal of mechanical science and technology* 25.11 (2011), pp. 2907–2917.

[31] H. Ahmad and T. Namerikawa. "Extended Kalman filter-based mobile robot localization with intermittent measurements". In: *Systems Science & Control Engineering: An Open Access Journal* 1.1 (2013), pp. 113–126.

[32] L. C. Carrillo-Arce et al. "Decentralized multi-robot cooperative localization using covariance intersection". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 1412–1417.

[33] H. Li and F. Nashashibi. "Cooperative multi-vehicle localization using split covariance intersection filter". In: *IEEE Intelligent transportation systems magazine* 5.2 (2013), pp. 33–44.

[34] D. Fox et al. "Particle filters for mobile robot localization". In: *Sequential Monte Carlo methods in practice* 499 (2001), p. 516.

[35] M. Rudafshani and S. Datta. "Localization in wireless sensor networks". In: *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. IEEE. 2007, pp. 51–60.

[36] L. Hu and D. Evans. "Localization for mobile sensor networks". In: *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM. 2004, pp. 45–57.

[37] B. Dil, S. Dulman, and P. Havinga. "Range-based localization in mobile sensor networks". In: *Wireless Sensor Networks*. Springer, 2006, pp. 164–179.

[38] J. Yi, S. Yang, and H. Cha. "Multi-hop-based monte carlo localization for mobile sensor networks". In: *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE. 2007, pp. 162–171.

[39] A. Baggio and K. Langendoen. "Monte Carlo localization for mobile wireless sensor networks". In: *Ad Hoc Networks* 6.5 (2008), pp. 718–733.

[40] E. Stevens-Navarro, V. Vivekanandan, and V. W. S. Wong. "Dual and mixture Monte Carlo localization algorithms for mobile wireless sensor networks". In: *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*. IEEE. 2007, pp. 4024–4028.

[41] J. Sheu, W. Hu, and J. Lin. "Distributed localization scheme for mobile sensor networks". In: *IEEE Transactions on Mobile Computing* 9.4 (2010), pp. 516–526.

[42] D. Fox et al. "A probabilistic approach to collaborative multi-robot localization". In: *Autonomous robots* 8.3 (2000), pp. 325–344.

[43] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[44] S. Zhang et al. "Accurate and Energy-Efficient Range-Free Localization for Mobile Sensor Networks". In: *IEEE Transactions on Mobile Computing* 9.6 (2010), pp. 897–910. ISSN: 1536-1233.

[45] S. Lenser and M. Veloso. "Sensor resetting localization for poorly modelled mobile robots". In: *IEEE International Conference on Robotics and Automation*. Vol. 2. 2000, pp. 1225–1232.

[46] S. Safavi and U. A. Khan. "Asymptotic Stability of LTV Systems With Applications to Distributed Dynamic Fusion". In: *IEEE Transactions on Automatic Control* 62.11 (2017), pp. 5888–5893. ISSN: 0018-9286. DOI: 10.1109/TAC.2017.2648501.

[47] S. Safavi and U. A. Khan. "Unbounded connectivity: Asymptotic stability criteria for stochastic LTV systems". In: *American Control Conference (ACC), 2016*. IEEE. 2016, pp. 7019–7024.

[48] S. Safavi and U. A. Khan. "Leader-follower consensus in mobile sensor networks". In: *IEEE Signal Processing Letters* 22.12 (2015), pp. 2249–2253.

[49] S. Safavi and U. A. Khan. "Dynamic leader-follower algorithms in mobile multi-agent networks". In: *Digital Signal Processing (DSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 10–13.

[50] S. Safavi and U. A. Khan. "A distributed range-based algorithm for localization in mobile networks". In: *Signals, Systems and Computers, 2016 50th Asilomar Conference on*. IEEE. 2016, pp. 1380–1384.

[51] S. Safavi and U. A. Khan. "On the convergence of time-varying fusion algorithms: Application to localization in dynamic networks". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 4907–4912. DOI: 10.1109/CDC.2016.7799019.

[52] S. Safavi and U. A. Khan. "An Opportunistic Linear–Convex Algorithm for Localization in Mobile Robot Networks". In: *IEEE Transactions on Robotics* 33.4 (2017), pp. 875–888. ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2686441.

[53] S. Safavi and U. A. Khan. "Localization in mobile networks via virtual convex hulls". In: *IEEE Transactions on Signal and Information Processing over Networks* PP.99 (2017), pp. 1–1. ISSN: 2373-776X. DOI: 10.1109/TSIPN.2017.2673238.

[54] S. Safavi et al. "A linear theory for distributed localization". In: *Proceedings of the IEEE* (2017). Manuscript submitted for publication.

[55] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. "Flocking in Fixed and Switching Networks". In: *IEEE Transactions on Automatic Control* 52.5 (2007), pp. 863–868.

[56] U. A. Khan, S. Kar, and J. M. F. Moura. "DILAND: An algorithm for distributed sensor localization with noisy distance measurements". In: *IEEE Transactions on Signal Processing* 58.3 (2010), pp. 1940–1947.

[57] M. M. Zavlanos and G. J. Pappas. "Dynamic Assignment in Distributed Motion Planning With Local Coordination". In: *IEEE Transactions on Robotics* 24.1 (2008), pp. 232–242.

[58] V. V. Kolpakov. "Matrix seminorms and related inequalities". In: *Journal of Mathematical Sciences* 23.1 (1983), pp. 2094–2106.

[59] G. C. Rota and W. Strang. "A note on the joint spectral radius". In: *Indagationes Mathematicae* 22 (1960), pp. 379–381.

[60] P. A. Parrilo and A. Jadbabaie. "Approximation of the joint spectral radius using sum of squares". In: *Linear Algebra and its Applications* 428.10 (2008), pp. 2385–2402.

[61] J. N. Tsitsiklis and V. D. Blondel. "The Lyapunov exponent and joint spectral radius of pairs of matrices are hard-when not impossible-to compute and to approximate". In: *Mathematics of Control, Signals and Systems* 10.1 (1997), pp. 31–40.

[62] G. Gripenberg. "Computing the joint spectral radius". In: *Linear Algebra and its Applications* 234 (1996), pp. 43–60.

[63] V. D. Blondel and Y. Nesterov. "Computationally efficient approximations of the joint spectral radius". In: *SIAM Journal on Matrix Analysis and Applications* 27.1 (2005), pp. 256–272.

[64] A. Jadbabaie, J. Lin, and A. S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on Automatic Control* 48.6 (2003), pp. 988–1001.

[65] W. Ren and R. W. Beard. "Consensus seeking in multiagent systems under dynamically changing interaction topologies". In: *IEEE Transactions on automatic control* 50.5 (2005), pp. 655–661.

[66] B. Touri and A. Nedic. "Product of Random Stochastic Matrices". In: *IEEE Transactions on Automatic Control* 59.2 (2014), pp. 437–448. ISSN: 0018-9286.

[67] J. Qin, H. Gao, and C. Yu. "On discrete-time convergence for general linear multi-agent systems under dynamic topology". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 1054–1059.

[68] K. Cai and H. Ishii. "Average consensus on arbitrary strongly connected digraphs with time-varying topologies". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 1066–1071.

[69] P.-Y Chevalier, J. M. Hendrickx, and R. M. Jungers. "Efficient algorithms for the consensus decision problem". In: *SIAM Journal on Control and Optimization* 53.5 (2015), pp. 3104–3119.

[70] G. Frana and J. Bento. "How is Distributed ADMM Affected by Network Topology?" In: *CoRR* abs/1710.00889 (2017). arXiv: 1710.00889. URL: http://arxiv.org/abs/1710.00889.

[71] S. M. Guu and C. T. Pang. "On the convergence to zero of infinite products of interval matrices". In: *SIAM journal on matrix analysis and applications* 25.3 (2003), pp. 739–751.

[72] D. J. Hartfiel. "On infinite products of nonnegative matrices". In: *SIAM Journal on Applied Mathematics* 26.2 (1974), pp. 297–301.

[73] N. Pullman. "Infinite products of substochastic matrices". In: *Pacific Journal of Mathematics* 16.3 (1966), pp. 537–544.

[74] L. Elsner and S. Friedland. "Norm Conditions for Convergence of infinite products". In: *Linear algebra and its applications* 250 (1997), pp. 133–142.

[75] S. Safavi and U. A. Khan. "Asymptotic stability of stochastic LTV systems with applications to distributed dynamic fusion". In: *CoRR* abs/1412.8018 (2014). arXiv: 1412.8018. URL: http://arxiv.org/abs/1412.8018.

[76] S. Safavi and U. A. Khan. "Localization and tracking in mobile networks: Virtual convex hulls and beyond". In: *CoRR* abs/1504.01207 (2015). arXiv: 1504.01207. URL: http://arxiv.org/abs/1504.01207.

[77] H. Anton. *Elementary linear algebra*. 10th ed. Hoboken, NJ: John Wiley & Sons, 2010.

[78] A. Ghaffarkhah and Y. Mostofi. "Path Planning for Networked Robotic Surveillance". In: *IEEE Transactions on Signal Processing* 60.7 (2012), pp. 3560–3575.

[79] F. Mourad et al. "Anchor-Based Localization via Interval Analysis for Mobile Ad-Hoc Sensor Networks". In: *IEEE Transactions on Signal Processing* 57.8 (2009), pp. 3226–3239.

[80] C. Kreucher, K. Kastella, and A. Hero. "Sensor management using an active sensing approach". In: *IEEE Transactions on Signal Processing* 85.3 (2005), pp. 607–624.

[81] R. Rahman, M. Alanyali, and V. Saligrama. "Distributed tracking in multihop sensor networks with communication delays". In: *IEEE Transactions on Signal Processing* 55.9 (2007), pp. 4656–4668.

[82] S. A. Aldosari and J. M. F. Moura. "Distributed detection in sensor networks: Connectivity graph and small world networks". In: *The 39th Asilomar Conference on Signals, Systems, and Computers*. 2005.

[83] S. Kar, S. Aldosari, and J. M. F. Moura. "Topology for Distributed Inference on Graphs". In: *IEEE Transactions on Signal Processing* 56.6 (2008), pp. 2609–2613.

[84] P. Braca et al. "Asymptotic optimality of running consensus in testing binary hypotheses". In: *IEEE Transactions on Signal Processing* 58.2 (2010), pp. 814–825.

[85] U. A. Khan and J. M. F. Moura. "Distributing the Kalman filter for large-scale systems". In: *IEEE Transactions on Signal Processing* 56(1).10 (2008), pp. 4919–4935.

[86] I. D. Schizas, A. Ribeiro, and G. B. Giannakis. "Consensus in Ad Hoc WSNs with Noisy Links - Part I: Distributed Estimation of Deterministic Signals". In: *IEEE Transactions on Signal Processing* 56.1 (2008), pp. 350–364.

[87] C. G. Lopes and A. H. Sayed. "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis". In: *IEEE Transactions on Signal Processing* 56.7 (2008), pp. 3122–3136.

[88] S. Safavi and U. A. Khan. "Revisiting finite-time distributed algorithms via successive nulling of eigenvalues". In: *IEEE Signal Processing Letters* 22.1 (2015), pp. 54–57.

[89] R. Olfati-Saber and R. M. Murray. "Consensus problems in networks of agents with switching topology and time-delays". In: *IEEE Transactions on Automatic Control* 49.9 (2004), pp. 1520–1533.

[90] S. Kar and J. M. F. Moura. "Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise". In: *IEEE Transactions on Signal Processing* 57.1 (2009), pp. 355–369.

[91] T. C. Aysal, M. J. Coates, and M. G. Rabbat. "Distributed average consensus with dithered quantization". In: *IEEE Transactions on Signal Processing* 56.10 (2008), pp. 4905–4918.

[92] T. C. Aysal, B. N. Oreshkin, and M. J. Coates. "Accelerated Distributed Average Consensus via Localized Node State Prediction". In: *IEEE Transactions on Signal Processing* 57.4 (2009), pp. 1563–1576.

[93] E. Kokiopoulou and P. Frossard. "Polynomial filtering for fast convergence in distributed consensus". In: *IEEE Transactions on Signal Processing* 57.1 (2009), pp. 342–354.

[94] A. Olshevsky and J. N. Tsitsiklis. "Convergence speed in distributed consensus and averaging". In: *SIAM review* 53.4 (2011), pp. 747–772.

[95] U. A. Khan, S. Kar, and J. M. F. Moura. "Distributed sensor localization in random environments using minimal number of anchor nodes". In: *IEEE Transactions on Signal Processing* 57.5 (2009), pp. 2000–2016.

[96] R. Olfati-Saber, J. A. Fax, and R. M. Murray. "Consensus and cooperation in networked multi-agent systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 215–233.

[97] W. Ren, R. W. Beard, and E. M. Atkins. "Information consensus in multivehicle cooperative control". In: *Control Systems, IEEE* 27.2 (2007), pp. 71–82.

[98] A. K. Das et al. "A vision-based formation control framework". In: *Robotics and Automation, IEEE Transactions on* 18.5 (2002), pp. 813–825.

[99] A. Jadbabaie, J. Lin, and A. S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on Automatic Control* 48.6 (2003), pp. 988–1001. ISSN: 0018-9286.

[100] C. W. Wu. "Agreement and consensus problems in groups of autonomous agents with linear dynamics". In: *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005.* IEEE. 2005, pp. 292–295.

[101] M. Cao, A. S. Morse, and B. D. O. Anderson. "Reaching a consensus in a dynamically changing environment: a graphical approach". In: *SIAM Journal on Control and Optimization* 47.2 (2008), pp. 575–600.

[102] D. Bajovic, J. Xavier, and B. Sinopoli. "Products of stochastic matrices: Exact rate for convergence in probability for directed networks". In: *Telecommunications Forum (TELFOR), 2012 20th.* IEEE. 2012, pp. 883–886.

[103] S. Boyd et al. "Randomized gossip algorithms". In: *Information Theory, IEEE Transactions on* 52.6 (2006), pp. 2508–2530.

[104] S. Safavi and U. A. Khan. "On the convergence rate of swap-collide algorithm for simple task assignment". In: *2014 48th Asilomar Conference on Signals, Systems and Computers.* 2014, pp. 1507–1510.

[105] D. Blatt and A. O. Hero. "Energy-based sensor network source localization via projection onto convex sets". In: *IEEE Transactions on Signal Processing* 54.9 (2006), pp. 3614–3619.

[106] U. A. Khan, S. Kar, and J. M. F. Moura. "Distributed algorithms in sensor networks". In: *Handbook on sensor and array processing*. Ed. by S. Haykin and K. J. Ray Liu. New York, NY: Wiley-Interscience, 2009.

[107] M. Koivisto et al. "Continuous Device Positioning and Synchronization in 5G Dense Networks with Skewed Clocks". In: *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2017.

[108] B. Bayat et al. "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques". In: *Current Opinion in Biotechnology* 45.Supplement C (2017). Energy biotechnology Environmental biotechnology, pp. 76 –84. ISSN: 0958-1669.

[109] T. Arampatzis, J. Lygeros, and S. Manesis. "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks". In: *Proceedings of the 2005 IEEE International Symposium on Mediterrean Conference on Control and Automation Intelligent Control, 2005.* 2005, pp. 719–724.

[110] N. Patwari et al. "Locating the nodes: cooperative localization in wireless sensor networks". In: *IEEE Signal processing magazine* 22.4 (2005), pp. 54–69.

[111] F. Gustafsson and F. Gunnarsson. "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available

wireless network measurements". In: *IEEE Signal processing magazine* 22.4 (2005), pp. 41–53.

[112] J. Albowicz, A. Chen, and L. Zhang. "Recursive position estimation in sensor networks". In: *Network Protocols, 2001. Ninth International Conference on*. IEEE. 2001, pp. 35–41.

[113] C. Savarese, J. M. Rabaey, and J. Beutel. "Location in distributed ad-hoc wireless sensor networks". In: *International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01).* Vol. 4. IEEE. 2001, pp. 2037–2040.

[114] R. L. Moses, D. Krishnamurthy, and R. M. Patterson. "A self-localization method for wireless sensor networks". In: *EURASIP Journal on Applied Signal Processing* (2003), pp. 348–358.

[115] G. Destino and G. Abreu. "On the maximum likelihood approach for source and network localization". In: *IEEE Transactions on Signal Processing* 59.10 (2011), pp. 4954–4970.

[116] N. Kantas, S. S. Singh, and A. Doucet. "Distributed maximum likelihood for simultaneous self-localization and tracking in sensor networks". In: *IEEE Transactions on Signal Processing* 60.10 (2012), pp. 5038–5047.

[117] Y. Shang and W. Ruml. "Improved MDS-based localization". In: *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE. 2004, pp. 2640–2651.

[118] P. Biswas et al. "Semidefinite programming based algorithms for sensor network localization". In: *ACM Transactions on Sensor Networks (TOSN)* 2.2 (2006), pp. 188–220.

[119] Y. Ding et al. "Sensor network localization, Euclidean distance matrix completions, and graph realization". In: *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*. ACM. 2008, pp. 129–134.

[120] M. Cao, B. D. O. Anderson, and A. S. Morse. "Sensor network localization with imprecise distances". In: *Systems and Control Letters* 55.11 (2006), pp. 887 –893. ISSN: 0167-6911.

[121] V. Ramadurai and M. L. Sichitiu. "Localization in Wireless Sensor Networks: A Probabilistic Approach". In: *International conference on wireless networks*. 2003.

[122] R. Peng and M. L. Sichitiu. "Probabilistic localization for outdoor wireless sensor networks". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 11.1 (2007), pp. 53–64.

[123] R. Nagpal, H. Shrobe, and J. Bachrach. "Organizing a global coordinate system from local information on an Ad Hoc sensor network". In: *Information Processing in Sensor Networks*. Springer. 2003, pp. 333–348.

[124] Y. Zhou, J. Li, and L. Lamont. "Multilateration localization in the presence of anchor location uncertainties". In: *Global Communications Conference (GLOBECOM)*. IEEE. 2012, pp. 309–314.

[125] S. Lederer, Y. Wang, and J. Gao. "Connectivity-based localization of large-scale sensor networks with complex shape". In: *ACM Transactions on Sensor Networks (TOSN)* 5.4 (2009), p. 31.

[126] A. Stanoev et al. "Cooperative method for wireless sensor network localization". In: *Ad Hoc Networks* 40 (2016), pp. 61–72.

[127] G. Wang et al. "Sensor relocation in mobile sensor networks". In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. Vol. 4. IEEE. 2005, pp. 2302–2312.

[128] B. Liu et al. "Mobility improves coverage of sensor networks". In: *Proceedings of the 6th ACM international symposium on Mobile Ad Hoc networking and computing*. ACM. 2005, pp. 300–308.

[129] A. Oracevic and S. Ozdemir. "A survey of secure target tracking algorithms for wireless sensor networks". In: *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. IEEE. 2014, pp. 1–6.

[130] E. Borgia. "The Internet of Things vision: Key features, applications and open issues". In: *Computer Communications* 54 (2014), pp. 1–31.

[131] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[132] H. Ghayvat et al. "WSN-and IOT-based smart homes and their extension to smart buildings". In: *Sensors* 15.5 (2015), pp. 10350–10379.

[133] S. Zhang et al. "Locating nodes in mobile sensor networks more accurately and faster". In: *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*. IEEE. 2008, pp. 37–45.

[134] I. Amundson and X. D. Koutsoukos. "A survey on localization for mobile wireless sensor networks". In: *Mobile Entity Localization and Tracking in GPS-less Environnments*. Springer, 2009, pp. 235–254.

[135] I. F. Akyildiz et al. "Wireless sensor networks: a survey". In: *Computer networks* 38.4 (2002), pp. 393–422.

[136] D. Niculescu and B. Nath. "DV based positioning in ad hoc networks". In: *Telecommunication Systems* 22.1 (2003), pp. 267–280.

[137] L. Lazos and R. Poovendran. "SeRLoc: Secure range-independent localization for wireless sensor networks". In: *Proceedings of the 3rd ACM workshop on Wireless security*. ACM. 2004, pp. 21–30.

[138] T. He et al. "Range-free localization schemes for large scale sensor networks". In: *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM. 2003, pp. 81–95.

[139] T. Camp, J. Boleng, and V. Davies. "A survey of mobility models for ad hoc network research". In: *Wireless Communications and Mobile Computing* 2.5 (2002), pp. 483–502. ISSN: 1530-8677. DOI: 10.1002/wcm.72.

[140] C. Perkins, E. Belding-Royer, and S. Das. *Ad Hoc on-demand distance vector (AODV) routing*. Tech. rep. RFC 3561. IETF, 2003.

[141] C. Schindelhauer. "Mobility in wireless networks". In: *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer. 2006, pp. 100–116.

[142] X. Hong et al. "A group mobility model for Ad Hoc wireless networks". In: *Proceedings of the 2nd ACM international workshop on Modeling,*

analysis and simulation of wireless and mobile systems. ACM. 1999, pp. 53–60.

[143] J. Harri, F. Filali, and C. Bonnet. "Mobility models for vehicular Ad Hoc networks: A survey and taxonomy". In: *IEEE Communications Surveys and Tutorials* 11.4 (2009).

[144] M. J. Sippl and H. A. Scheraga. "Cayley-Menger coordinates". In: *Proceedings of the National Academy of Sciences* 83.8 (1986), pp. 2283–2287.

[145] A. F. Möbius. *Der barycentrische calcul*. 1827.

[146] N. Patwari. "Location Estimation in Sensor Networks". Ph.D. University of Michigan–Ann Arbor, 2005.

[147] Y. G. Kim, J. An, and K. D. Lee. "Localization of Mobile Robot Based on Fusion of Artificial Landmark and RF TDOA Distance under Indoor Sensor Network". In: *International Journal of Advanced Robotic Systems* 8.4 (2011), pp. 203–211.

[148] U. A. Khan, S. Kar, and J. M. F. Moura. "Linear theory for self-localization: Convexity, barycentric coordinates, and Cayley–Menger determinants". In: *IEEE Access* 3 (2015), pp. 1326–1339.

[149] D. Niculescu and B. Nath. "Ad hoc positioning system (APS) using AOA". In: *INFOCOM 2003. Twenty-Second Annual Joint Conference*

of the IEEE Computer and Communications. IEEE Societies. Vol. 3. Ieee. 2003, pp. 1734–1743.

[150] J. Liu, Y. Zhang, and F. Zhao. "Robust distributed node localization with error management". In: *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. ACM. 2006, pp. 250–261.

[151] F. Thomas and L. Ros. "Revisiting trilateration for robot localization". In: *Robotics, IEEE Transactions on* 21.1 (2005), pp. 93–101.

[152] P. M. Maxim et al. "Trilateration Localization for Multi-robot Teams." In: *ICINCO-RA (2)*. 2008, pp. 301–307.

[153] Y. Zhou. "An efficient least-squares trilateration algorithm for mobile robot localization". In: *IEEE International Conference on Intelligent Robots and Systems*. 2009, pp. 3474–3479.

[154] L. E. Navarro-Serment, C. J. J. Paredis, and P. K. Khosla. "A beacon system for the localization of distributed robotic teams". In: *Proceedings of the International Conference on Field and Service Robotics*. Vol. 6. 1999.

[155] K. S. Chong and L. Kleeman. "Accurate odometry and error modelling for a mobile robot". In: *IEEE International Conference on Robotics and Automation, 1997. Proceedings*. Vol. 4. IEEE. 1997, pp. 2783–2788.

[156] T. R. Wanasinghe, G. K. I. Mann, and R. G. Gosine. "Decentralized cooperative localization for heterogeneous multi-robot system using split covariance intersection filter". In: *2014 Canadian Conference on Computer and Robot Vision (CRV)*. IEEE. 2014, pp. 167–174.

[157] L. Paull, M. Seto, and J. J. Leonard. "Decentralized cooperative trajectory estimation for autonomous underwater vehicles". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 184–191.

[158] L. C. Carrillo-Arce et al. "Decentralized multi-robot cooperative localization using covariance intersection". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1412–1417.

[159] H. M. Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[160] A. Boukerche et al. "Localization systems for wireless sensor networks". In: *Wireless Communications, IEEE* 14.6 (2007), pp. 6–12. ISSN: 1536-1284.

[161] W. Wang and Q. Zhu. "Sequential Monte Carlo localization in mobile sensor networks". In: *Wireless Networks* 15.4 (2009), pp. 481–495. ISSN: 1572-8196.

[162] S. A. Mageid. "Autonomous localization scheme for mobile sensor networks in fading environments". In: *IEEE International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT)*. 2016, pp. 1–8.

[163] V. Fox et al. "Bayesian filtering for location estimation". In: *IEEE Pervasive Computing* 2.3 (2003), pp. 24–33.

[164] M. S. Arulampalam et al. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 174–188.