Distributed Estimation and Control for Robotic Networks

Andrea Simonetto

Ph.D. Thesis

DISTRIBUTED ESTIMATION AND CONTROL FOR ROBOTIC NETWORKS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

maandag 5 november 2012 om 10 uur

door

Andrea SIMONETTO

Master of Science in Space Engineering
Politecnico di Milano and Politecnico di Torino
geboren te Cantù, Italië

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. R. Babuška, M.Sc.

Copromotor: Dr. ir. T. Keviczky

This dissertation has been completed in partial fulfillment of the requirements for the graduate study of the Dutch Institute of Systems and Control (DISC).

ISBN: . . .

Printed in The Netherlands
Author's email: ekkekkose@gmail.com

“There is no royal road to Geometry”

Εὐκλίδης *(Euclid)*

Acknowledgments

This thesis is the work of four very intense years of my life. I wish to express my gratitude to all the people who supported me and helped me make this happen.

First or all, I want to thank Tamás and Robert, my supervisors, for their patience in teaching me how to write papers in a catching but still rigorous way. For the freedom they gave me in seeking my research paths, also outside the Netherlands; for their constant supervision, when it was necessary. It was great to work with you both, and I hope we will continue doing it.

I want to thank Prof. Mikael Johansson, at KTH, for hosting me for three months in the welcoming Stockholm. It was nice to be in your "optimization" group, even if for a short time.

I want to thank all my committee members for reading the initial draft of the thesis and giving me precious feedback.

I wish to express my gratitude to the colleagues at DCSC, who shared with me many many long working days, some conferences, some basketball, and good times. Thanks Aleksandar, Alessandro, Amol, Arturo, Dang, Hong Song, İlhan, Ilya, Ivo, Ivo, Jacopo, Lucian, Noor, Samira, Solomon, Stefan, Zhe, ZSófia and the last arrived Dario. Thanks Aydın and Paweł for sharing an apartment with me, for the talks, and for trying to educate me on Turkish and Polish traditions. And especially, thanks to have convinced me to live in Leiden. Thanks to Gijs, who translated the summary and the propositions in Dutch, and Marco who helped me with the Italian version. Thanks to my officemates and laugh-mates Gabriel, Roland, and Coen. It was great fun to gossip with you.

A very special thank goes to Justin and Paolo. Thanks for teaching me how to do a PhD in the Netherlands. Thanks for being friends before colleagues. Thanks Paolo for your incredible style. Thanks Justin for your way of living. Thanks both for showing me that even Italians can cook chinese, and even Americans can make superb Pizzas. I hope our paths will cross again.

I want to thank my master's and bachelor students who always had the ability to make me smile, and who taught me what really means being a student at TU Delft. Thanks Daniele, Mehdi, Thomas, and Wessel for your great enthusiasm.

Thanks to the Stockholm crew. Magnus, thanks for sharing the office and explaining me your Swedish way. Thanks to Euhanna and Burak for the discussions, going-out, making me feel welcome even far from Delft. Thanks Dimos, for the guidance. And, thanks to all of you, for making it extremely pleasant.

Thanks to my friends, the ones who shared with me the passion (in the Gospel sense) of the Dutch language, the ones who share with me the Rock'n Delft association and activities, and all the rest. Thanks Vincent, Emanuele, Cristian, Sonia, Thanos (ela re!), Eleni. Thanks Jules, Jeroen, Fien, Patricio, Federico and all the other dancing enthusiasts. Thanks Gianni, Tungky, Lihn. Thanks to Rodrigo, a good guy, a guidance, a couch to crash on, a friend of many beers, spareribs, and endless talks.

Grazie ai miei amici ancora in Italia. Grazie per parlarmi ancora, anche se non torno quasi mai. Grazie a Ricky, Ale, Matteo, Mauro, per esserci venuti almeno una volta, qui nel paese delle piogge. Siete stati sicuramente coraggiosi. Grazie a Guido, per farmi vedere come uno può fare mille cose insieme, tutte al massimo livello. Come uno può fare il dottorato e allo stesso tempo lavorare. Grazie per la tua visione pratica, bresciana, della vita. Grazie a Roberto per essermi amico da una vita, per esserci quando se ne ha bisogno e per le lunghe, infinite, chiacchierate in mansarda. Grazie Bob.

Grazie ai miei genitori, che hanno accettato, non certo a cuor leggero, di vedermi andare all'estero per inseguire le mie passioni. Grazie per il loro essere dei grandi genitori. Grazie alle nonne, che non mi hanno ancora perdonato di non tornare quasi mai, e agli zii.

Finally thanks to the two girls of my life. La mia sisterella et Hélène. Grazie Tizzi per esserci nella buona e cattiva sorte. Per raccontarmi la tua vita, per ridere sui film stupidi e per essere venuta più e più volte a trovarmi. I tuoi capelli non te l'hanno perdonato, ma io ne ero sempre felice. Merci Hélène pour être ce que tu es: exceptionnelle. Merci pour ta compréhension quand j'étais loin ou étais tout le temps en train de travailler sur ma thèse. Merci pour m'avoir montré que si on est très motivé, on peut même organizer des courses du Rock à Delft. Et surtout, merci pour m'avoir donné une raison de finir ce doctorat.

Delft, September 2012
Andrea Simonetto

Contents

Chapter 1

Introduction

Groups of sensors and autonomous mobile robots that exchange information with one another are envisioned to play an important role in several societally relevant applications. Examples range from monitoring and surveillance, tracking, exploration to search, rescue, and disaster relief. More specific applications encompass forest fire monitoring with multiple unmanned aerial vehicles (Casbeer et al., 2005), coordinated control of multiple underwater robots (Leonard et al., 2010), earthquake predictions and damage assessment (Chaamwe et al., 2010, Oguni et al., 2011), deep space exploration (Izzo and Pettazzi, 2007), and robot-assisted search and rescue in response to natural disasters or other calamities (Casper and Murphy, 2003).

In all these applications, the communication among the different sensors and robots is crucial in order to accomplish the mission tasks. This gives rise to a communication network that describes the way the sensors and robots communicate with each other. In this context, if two sensors or robots can communicate with one another there is a link between them in the network. Extending this terminology, the group of sensors communicating with one another via a communication network are typically referred to as sensor network, while the group of robots are sometimes called mobile robotic networks. In some situations, such as the case of mobile robots carrying sensors onboard, the distinction among sensor and mobile robotic networks can be less immediate. As a result we often use in this thesis the term *robotic networks* to identify either sensor networks, mobile robotic networks, or a combination of both.

From a theoretical and implementation perspective, the studies of robotic networks involve distributed estimation, control, and optimization, which all include the design of distributed algorithms. The word “distributed” indicates here the adaptation of the standard concepts of estimation, control, and optimization to settings where the sensors or the robots are endowed with local information processing/computation capabilities, have a local knowledge of the environment and of the entire group, and they need to communicate with one another to achieve the common estimation, control, or optimization objective.

The challenges that robotic networks and the design of distributed algorithms pose are diverse. Some of the most important ones are linked to the changing nature and limitations of the networks. Both sensors and robots have to be able to cope with (unexpected) variations of the communication topology. The algorithms need to be suitable for the limited computation and communication capabilities of the sensors and the robots, i.e., we cannot expect to have access to unlimited computation and communication power onboard. Moreover, there is the need of formal guarantees for the algorithms to achieve given performances, for example guarantees that the robots do not collide with each another and that the communication network maintains a certain connectivity.

Among the diverse challenges, in this thesis we will consider specific aspects related to the following:

- Distributed estimation algorithms for sensor networks have been applied mainly on sensors that observe linear time-invariant systems. Nonetheless, there are many realistic situations in which such a framework cannot be applied, due to nonlinearities in the dynamical system, the sensing equation, or due to the presence of constraints. One typical application example where all of these complicating characteristics are present is the localization of a moving object via range-only measurements.

- Sensor networks are comprised of many and possibly different sensors with their own capabilities. However, with current algorithms it is not possible to tailor the computational effort to the computational capabilities of the sensors, which prevents utilizing the full potential of the sensor network.

- Considering distributed control algorithms, the connectivity of the communication network among the mobile robots has been often considered to be granted by assumption, rather than being achieved as an objective of the distributed control action. In addition, the connectivity has often been analyzed as a binary statement (i.e., is it connected? yes/no), whereas more useful insights could be obtained by describing the connectivity by a suitable continuous metric. This metric could express how well connected the network is and how we could increase the connectivity (for example, by moving the robots closer to each other).

- Distributed optimization algorithms are typically based on slow converging iterative schemes that can guarantee feasibility of the solution with respect to the constraints only at convergence (i.e., asymptotically). This might endanger the physical implementability of the algorithms on real hardware due to the limited communication capabilities of the sensors and robots.

1.1 Objective and Outline

The main objective of this thesis is to analyze how we can make distributed estimation, control, and optimization techniques more suitable for robotic networks. In particular, we will propose methods to tackle the specific aspects presented in the previous section, i.e., nonlinearities, heterogeneity, connectivity of the communication network, feasibility of the solutions, and real-time implementation.

This thesis consists of six chapters. Chapter 1 is this introduction, Chapter 2, 3, 4, and 5 form the main material of the thesis, while Chapter 6 gives our conclusions and recommendations. With regards to the main chapters,

Chapter 2 deals with distributed nonlinear estimation. In this chapter we consider sensor networks that are used to estimate the state of a given nonlinear dynamic process, such as the state of a mobile robot.

Chapter 3 analyzes the distributed/parallel computation side of nonlinear estimation. In particular, in this chapter we consider networks of computing units (the cores of a GPU-architecture).

Chapter 4 studies distributed control solutions for networks whose links are weighted via the pair-wise distances of the nodes. These nodes represent mobile robots and therefore the connectivity of the network depends on the robots' position (i.e., we consider state-dependent graphs). In this chapter we formulate and solve local control problems that aim to move the robots in order to increase the algebraic connectivity of their interconnecting communication network. This problem is then extended to jointly optimize the connectivity of the communication graph and the connectivity of the robots with a number of moving targets.

Chapter 5 investigates convex and non-convex networked optimization problems with resource allocation constraints, which can be applied to various robotic network applications. In this chapter we consider networks of computing units, either mobile or non-moving.

1.2 Contributions

The following are the main contributions of the thesis.

Chapter 2: Distributed Nonlinear State Estimation

- We propose a unified framework for the distributed nonlinear estimation problem. In our framework the nonlinearities are handled locally by the sensor nodes, while a weighted merging mechanism provides a method to incorporate the information coming from the neighboring sensor nodes. This mechanism allows the use of different estimators on different sensor nodes.

- We propose a distributed nonlinear Moving Horizon Estimator.

- We propose distributed versions of commonly used nonlinear estimators, i.e., Particle Filters, Unscented and Extended Kalman Filters. These estimators are observed to lead to a better estimation quality than the ones available in the literature in numerical simulations. This improvement is due to the underlying weighted merging mechanism.

The results contained in this chapter have been submitted or published in

Simonetto and Keviczky (2012) A. Simonetto and T. Keviczky. *Distributed Decision Making and Control*, volume 417 of *Lecture Notes in Control and Information Sciences*, chapter Distributed Nonlinear Estimation for Diverse Sensor Devices, pages 147 – 169. Springer, 2012.

Simonetto et al. (2011a) A. Simonetto, D. Balzaretti, and T. Keviczky. Evaluation of a Distributed Moving Horizion Estimator for a Mobile Robot Localization Problem. In *Proceedings of the 18th IFAC World Congress*, pages 8902 – 8907, Milan, Italy, August – September 2011a.

Simonetto et al. (2010a) A. Simonetto, T. Keviczky, and R. Babuška. Distributed Nonlinear Estimation for Robot Localization using Weighted Consensus. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3026 – 3031, Anchorage, USA, May 2010a.

Simonetto et al. A. Simonetto, T. Keviczky, and R. Babuška. Distributed Nonlinear State Estimation of Mobile Robots via Sensor Networks . In preparation: to be submitted as Springer Brief, 2012.

Chapter 3: Distributed Computation Particle Filters on GPU-Architectures

- We analyze how to distribute the computations of Particle Filters among different computing units and we devise an algorithm that can achieve accurate estimation results, while being implemented in real-time.
- We implement the resulting distributed computation Particle Filter on a robotic arm experimental setup using parallel GPU-architectures, where we use the result of a Particle Filter based on over a million particles as an input for a real-time feedback controller with a sampling frequency of 100 Hz.

The results contained in this chapter have been submitted or published in

Chitchian et al. (2012a) M. Chitchian, A. Simonetto, A. S. van Amesfoort, and T. Keviczky. Distributed Computation Particle Filters on GPU-Architectures for Real-Time Control Applications. *Submitted to IEEE Transactions on Control Systems Technology*, 2012a.

Simonetto and Keviczky (2012) A. Simonetto and T. Keviczky. *Distributed Decision Making and Control*, volume 417 of *Lecture Notes in Control and Information Sciences*, chapter Distributed Nonlinear Estimation for Diverse Sensor Devices, pages 147 – 169. Springer, 2012.

Simonetto and Keviczky (2009) A. Simonetto and T. Keviczky. Recent Developments in Distributed Particle Filters: Towards Fast and Accurate Algorithms. In *Proceedings of the 1st IFAC Workshop on Estimation and Control of Networked Systems*, pages 138 – 143, Venice, Italy, September 2009.

Chapter 4: Distributed Control of Robotic Networks with State-Dependent Laplacians

- We extend and modify the standard centralized optimization procedure of (Kim and Mesbahi, 2006, Derenick et al., 2009) for the maximization of the algebraic connectivity (which is a measure of connectivity "quality"), in order to handle more realistic robot dynamics. The resulting optimization problem is then proven to be feasible at each discrete time step under rather general assumptions.
- We propose a distributed solution for the resulting centralized problem. Our distributed approach relies on local problems that are solved by each robot using information from nearby neighbors only and, in contrast to (De Gennaro and Jadbabaie, 2006), it does not require any iterative schemes, making it more suitable for real-time applications.
- We extend the mentioned distributed solution to tackle the problem of collectively tracking a number of moving targets while maintaining a certain level of connectivity among the network of mobile robots.

The results contained in this chapter have been submitted or published in

Simonetto et al. (2012a) A. Simonetto, T. Keviczky, and R. Babuška. Constrained Distributed Algebraic Connectivity Maximization in Robotic Networks. *Submitted to Automatica*, 2012a.

Simonetto et al. (2013) A. Simonetto, T. Keviczky, and R. Babuška. *Distributed Autonomous Robotic Systems*, volume 83 of *STAR*, chapter Distributed Algebraic Connectivity Maximization for Robotic Networks: A Heuristic Approach, pages 267 – 279. Spriger, 2013.

Simonetto and Keviczky (2011) A. Simonetto and T. Keviczky. Distributed Multi-Target Tracking via Mobile Robotic Networks: a Localized Non-iterative SDP Approach. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 4226 – 4231, Orlando, USA, December 2011.

Simonetto et al. (2011b) A. Simonetto, T. Keviczky, and R. Babuška. On Distributed Algebraic Connectivity Maximization in Robotic Networks. In *Proceedings of the American Control Conference*, pages 2180 – 2185, San Francisco, USA, June – July 2011b.

Simonetto et al. (2010b) A. Simonetto, T. Keviczky, and R. Babuška. Distributed Algebraic Connectivity Maximization for Robotic Networks: A Heuristic Approach. In *Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems*, Lausanne, Switzerland, November 2010b.

Chapter 5: Distributed Optimization Methods in Robotic Network Applications [1]

- We propose a regularized saddle-point algorithm for convex networked optimization problems with resource allocation constraints. Our approach ensures that each iterative update step satisfies the resource allocation constraints and makes the scheme faster than traditional subgradient algorithms. Furthermore, we demonstrate the relevance of the scheme in a representative robotic scenario.

- We solve a particular non-convex networked optimization problem, known as the Maximum Variance Unfolding problem and its dual, the Fastest Mixing Markov Process problem with the same distributed primal-dual subgradient iterations. The convergence of our method is proven even in the case of approximation errors in the calculation of the subgradients. Finally, we illustrate the importance of these problems in robotic networks as formulation of localization problems and coverage (or dispersion) control.

Part of the results contained in this chapter have been submitted to

Simonetto et al. (2012c) A. Simonetto, T. Keviczky, and M. Johansson. A Regularized Saddle-Point Algorithm for Networked Optimization with Resource Allocation Constraints. 2012c. *To be presented at the 51st IEEE Conference on Decision and Control*, Maui, USA, December 2012.

Simonetto et al. (2012b) A. Simonetto, T. Keviczky, and D.V. Dimarogonas. Distributed Solution for a Maximum Variance Unfolding Problem with Sensor and Robotic Network Applications. 2012b. *Presented at the 50th Allerton Conference*, Allerton, USA, October 2012.

[1] Part of the results of this chapter have been obtained during a three-month visit at KTH, The Royal Institute of Technology in Stockholm, Sweden, under the supervision of Prof. M. Johansson and Dr. D. V. Dimarogonas.

Chapter 2

Distributed Nonlinear State Estimation

***Abstract* —** In this chapter we consider the nonlinear state estimation problem via sensor networks, which is relevant both from a theoretical and an application perspective.

We present a unified way of describing distributed implementations of four commonly used nonlinear estimators: the Moving Horizon Estimator, the Particle Filter, the Extended and Unscented Kalman Filter. Leveraging on the presented framework, we propose new distributed versions of these methods, in which the nonlinearities are locally managed by the various sensor nodes whereas the different estimates are merged based on a weighted average consensus process. We show how the merging mechanism can handle different filtering algorithms implemented on heterogeneous sensors, which is especially useful when they are endowed with diverse local computational capabilities. Simulation results assess the performance of the algorithms with respect to standard distributed and centralized estimators.

2.1 Introduction

Nowadays, wireless sensor networks are developed to provide fast, cheap, reliable, and scalable hardware solutions to a large number of industrial applications, ranging from surveillance (Biswas and Phoha, 2006, Räty, 2010) and tracking (Songhwai et al., 2007, Liu et al., 2007) to exploration (Sun et al., 2005, Leonard et al., 2007), monitoring (Corke et al., 2010, Sun et al., 2011), and other sensing tasks (Arampatzis et al., 2005). From the software perspective, an increasing effort is spent on designing distributed algorithms that can be embedded in these sensor networks, providing high reliability with limited computation and communication requirements for the sensor nodes.

In this chapter we focus on proposing distributed methods for nonlinear state estimation using such sensor networks in a distributed sensing setting, where each sensor node has access to local measurements and can share data via the underlying network.

As expressed in Chapter 1, our motivations are twofold. First of all, from a theoretical point of view, distributed nonlinear estimators are in their early development stage and the challenges they pose are far from being solved. Second, from an application perspective, many real-life tasks ask for reliable, scalable, and distributable software to be embedded in sensor networks for nonlinear estimation purposes.

In this context, in Section 2.2 we formulate the distributed estimation problem and we propose a common framework where to develop the distributed estimators. This common framework is based on a merging mechanism that can also handle different classes of estimators implemented on the different sensor nodes. This is especially useful when the heterogeneous sensor devices have different computational capabilities and we want to exploit their resources efficiently. In this respect, the proposed merging mechanism can be used to tailor the composition of various filters to the diverse sensor devices in the network.

In Section 2.3 we leverage on the proposed framework and we design distributed versions of the most common nonlinear estimators. In particular, first we propose a distributed Moving Horizon Estimator that allows the most general assumptions on the system model and constraints to be treated in a rigorous, optimization-based framework. Then, restricting the generality of the assumptions on the system model and constraints, we propose versions of distributed Particle Filters and Unscented and Extended Kalman Filters.

Finally, numerical simulations illustrate the benefit of the common merging mechanism with respect to standard distributed algorithms and centralized estimators.

2.2 The Distributed Nonlinear Estimation Problem and Consensus Algorithms

2.2.1 Problem Formulation

Let the discrete-time nonlinear time-invariant dynamical model of the system with state $\mathbf{x}(k)$ be

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{w}(k)). \tag{2.1}$$

The state $\mathbf{x}$ and the disturbance $\mathbf{w}$ satisfy the constraints

$$\mathbf{x}(k) \in \mathbb{X} \subseteq \mathbb{R}^n \text{ and } \mathbf{w}(k) \in \mathbb{W} \subseteq \mathbb{R}^{\mathrm{w}}, \text{ for all } k, \tag{2.2}$$

where $\mathbb{X}$ and $\mathbb{W}$ are generic non-convex sets. The function $f : \mathbb{X} \times \mathbb{W} \to \mathbb{X}$ is a smooth nonlinear function and $0 \in \mathbb{W}$.

Let the process described in (2.1) be observed by N non-moving sensor nodes each with some processing and communication capability. Each of the sensor nodes has a copy of the nonlinear dynamical model (2.1). The sensor nodes are labeled $i = 1, \ldots, N$ and form the set $\mathcal{V}$. The sensor node communication topology is modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where an edge (i, j) is in $\mathcal{E}$ if and only if sensor node i and sensor node j can exchange messages. We assume the sensor nodes to have an unlimited sensing range, the graph to be connected, the sensor node clocks to be synchronized, and we assume perfect communication (no delays or packet losses).

The sensor nodes with which sensor node i communicates are called neighbors and are contained in the set $\mathcal{N}_i$. We define $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{i\}$ and $N_i^+ = |\mathcal{N}_i^+|$. Each sensor node i measures the quantity $\mathbf{z}_i(k)$, which is related to the state $\mathbf{x}(k)$ through the nonlinear measurement equation

$$\mathbf{z}_i(k) = g_i(\mathbf{x}(k)) + \boldsymbol{\mu}_i(k), \tag{2.3}$$

where $\boldsymbol{\mu}_i(k)$ is an additive noise term that satisfies

$$\boldsymbol{\mu}_i(k) \in \mathbb{M}_i \subseteq \mathbb{R}^{\mathrm{q}_i}, \quad 0 \in \mathbb{M}_i, \text{ for all } k, \tag{2.4}$$

and each of the $\mathbb{M}_i$ is a generic non-convex set, while each of the $g_i : \mathbb{X} \times \mathbb{M}_i \to \mathbb{X}$ is a smooth nonlinear function. The noise terms $\boldsymbol{\mu}_i(k)$ are assumed to be independent of each other, which is often a standard and reasonable assumption in practice. For simplicity, we will indicate with $\mathbf{z}(k)$ the stacked vector of all the measurements $\mathbf{z}_i(k)$, with $\boldsymbol{\mu}(k)$ the stacked vector of all the measurement noise $\boldsymbol{\mu}_i(k)$, while with $g(\mathbf{x}(k))$ we will denote the compacted stacked form of all the $g_i(\mathbf{x}(k))$, i.e.,

$$\mathbf{z}(k) = g(\mathbf{x}(k)) + \boldsymbol{\mu}(k). \tag{2.5}$$

We assume that the process described in (2.1) equipped with the stacked measurement equation (2.5) is strongly locally observable for all $\mathbf{x} \in \mathbb{X}$, meaning that the following map

$$O(\mathbf{x}) = \left(g(\mathbf{x}),\ g(f(\mathbf{x},0)),\ g(f(f(\mathbf{x},0),0)),\ \ldots,\ g(\underbrace{f(\cdots f(f}_{n-1}(\mathbf{x},0),0))) \right) \tag{2.6}$$

has rank n for all $\mathbf{x} \in \mathbb{X}$, (Nijmeijer, 1982, Albertini and D'Alessandro, 1995). This assumption implies that we can reconstruct the state of (2.1) at the discrete time k via the measurements $\mathbf{z}(k)$. We remark that the rank condition (2.6) is the nonlinear extension of the standard rank condition for linear systems.

In this chapter, we are interested in situations in which the process described in (2.1) is not strongly locally observable by the individual sensor nodes alone, meaning that the local couple $(\mathbf{z}_i(k), g_i)$ together with the dynamical model f is not sufficient to estimate the state $\mathbf{x}(k)$. More formally, we are interested in situation in which it is *not assumed* that the nodal observability map

$$O_i(\mathbf{x}) = \left(g_i(\mathbf{x}),\ g_i(f(\mathbf{x},0)),\ g_i(f(f(\mathbf{x},0),0)),\ \ldots,\ g_i(\underbrace{f(\cdots f(f}_{n-1}(\mathbf{x},0),0))) \right) \tag{2.7}$$

has rank n for all $\mathbf{x} \in \mathbb{X}$. Under this circumstance, each of the sensor nodes needs to communicate with the neighboring nodes to obtain their local couples $(\mathbf{z}_j(k), g_j)$ and, possibly, the ones of further away sensor nodes via consecutive and multi-hop communication.

We assume that after a *limited* amount of multi-hop communication, the nodal observability maps, extended with the information coming from the neighboring nodes, become full rank for all $\mathbf{x} \in \mathbb{X}$ and therefore each sensor node can estimate the state. Let $\bar{\mathbf{x}}_i(k)$ denote the estimate of sensor node i at time k. This local estimate $\bar{\mathbf{x}}_i(k)$ is in general a stochastic variable, thus we let $\mathbb{E}[\bar{\mathbf{x}}_i(k)]$ be its expected value, while $\bar{P}_i$ represents its covariance.

Since communication is an important resource in sensor networks, the sensor nodes will obtain only the data necessary to make the state observable (i.e., the nodal observability maps full rank), and not the whole network measurements. In addition, we remark that it is also beneficial to limit the number of measurements available to each sensor node. This, in order to keep the estimation problem small in size, thus easier to handle within the sensor nodes' limited computation resources. As a result, the nodal estimates $\bar{\mathbf{x}}_i(k)$ are in general different among each other. For this reason the sensor nodes can decide to communicate further to reduce this difference (and increase the estimation quality) and eventually agree on a common value for $\bar{\mathbf{x}}_i(k)$. Let τ be the total number of communication rounds each sensor node performs among its neighborhood before the subsequent time step $k+1$. Let $\hat{\mathbf{x}}_i(k,\tau)$ be the agreed value for the state estimate after τ round of communication (which for finite τ could be still different among the sensor nodes)[1]. The distributed estimation problem can be formulated, for each sensor nodes i, as follows.

Problem 2.1 Distributed Estimation Problem *Compute, on each sensor node i, the local estimate $\hat{\mathbf{x}}_i(k,\tau)$ of the state governed by the dynamical equation* (2.1) *making use of local measurements* (2.3) *and communication within the neighborhood $\mathcal{N}_i$. This local estimate $\hat{\mathbf{x}}_i(k,\tau)$ must:*

(i) satisfy the constraints on the state and noise terms, Equations (2.2) *and* (2.4) *for a given $\tau \ll \infty$;*

(ii) be an unbiased estimate for $\mathbf{x}(k)$, i.e., $\mathbb{E}[\hat{\mathbf{x}}_i(k,\tau)] = \mathbf{x}(k)$ for a given $\tau \ll \infty$;

(iii) converge, for $\tau \to \infty$, to a collective estimate that is the same for all the sensor nodes, i.e., $\lim_{\tau\to\infty} \hat{\mathbf{x}}_i(k,\tau) = \hat{\mathbf{x}}(k)$, for each $i \in \mathcal{V}$.

We note that, if we allow $\tau \to \infty$, it would be straightforward to solve Distributed Estimation Problem 2.1. In fact, it would be sufficient to communicate the sensor nodes data throughout the whole network. On the contrary, the main challenge in Problem 2.1 is to ensure requirements *(i)-(ii)* for a given $\tau \ll \infty$, typically $\tau = 1$ (meaning communication only with the neighborhood). This formally translates the sensor nodes communication limitation.

We remark that

- Requirement *(iii)* does not imply requirement *(ii)* nor vice-versa: in fact, in *(iii)* we only require the sensor nodes to agree on a common estimate asymptotically (in fact, this common value could be biased), while point *(ii)* requires the sensor nodes to deliver possibly different unbiased estimates at each time step k, i.e., $\mathbb{E}[\hat{\mathbf{x}}_i(k,\tau)] = \mathbb{E}[\hat{\mathbf{x}}_j(k,\tau)] = \mathbf{x}(k)$, but it can be that $\hat{\mathbf{x}}_i(k,\tau) \neq \hat{\mathbf{x}}_j(k,\tau)$, for each i and j (even for $\tau \to \infty$).
- If both requirements *(ii)* and *(iii)* are satisfied, then the sensor nodes agree asymptotically on an unbiased estimate for $\mathbf{x}(k)$.

In the next sections we propose different estimators that are specifically designed to tackle the Distributed Estimation Problem 2.1 and we will highlight the satisfaction of the requirements *(i)-(iii)*.

[1] We remark that, by definition, $\hat{\mathbf{x}}_i(k,0) = \bar{\mathbf{x}}_i(k)$.

Remark 2.1 (Graph topology). *In this chapter we make the simplifying assumption that the communication graph of the sensor network is fixed. For completeness we refer the reader to the works in (Xiao et al., 2005, 2006, Boyd et al., 2006, Fagnani and Zampieri, 2008), which deal with time-varying topology. In our opinion, this time-invariant graph assumption is not overly restrictive and we anticipate it could be removed by minor modifications of the methods to be presented.*

2.2.2 Motivations and Challenges

Before describing the estimators, we will elaborate on the underlying motivations and challenges of the Distributed Estimation Problem 2.1. First, recall the following motivations:

- There are many scenarios, especially in robotics, where nonlinear dynamics, nonlinear measurement equations, and constraints are present. Distributed Estimation Problem 2.1 is a natural extension of common problems described in the linear setting for sensor networks.
- There is an increasing number of applications where a large number of sensor nodes are employed to deliver reliable estimates for a common underlying process. In these applications the need for distributed operations comes directly from the nature and number of the sensors. In fact, given their number (some application are aiming at deploying 1000 or more sensors) we cannot expect to collect their measurements in only one computing unit which will have to deal with a large-scale nonlinear estimation problem. On the contrary, the individual sensor nodes will be required to perform local estimation and to communicate with the neighboring nodes in order to increase the estimation quality (and, in some cases, to make the process observable).

On challenges' side, we recall the following two points. The first main challenge is that the process and the measurement equation are nonlinear. Nonlinearity makes the estimation problem harder to solve computation-wise. Moreover, multiple solutions are often introduced in the nonlinear estimation process and issues linked to stability and bias can depend critically on the initial conditions.

The distributed nature of the problem is the second main challenge. Distribution introduces couplings among the different sensor nodes: the estimates are shared and combined together. This could damage the stability and unbiasedness properties of the local nonlinear estimators. Furthermore, trade-offs have to be made between communication, computational efforts and estimation quality. Sometimes, especially in the nonlinear setting, just one more round of communication (e.g., $\tau = 2$ instead of $\tau = 1$), could increase substantially both the computational time and the quality of the estimate.

With these motivations and challenges in mind, we start in the next section to consider distributed estimators.

2.2.3 General Framework for the Distributed Estimators

We propose to leverage on the same underlying framework for the distributed estimators we will design to solve Problem 2.1. This framework is depicted in Figure 2.1.

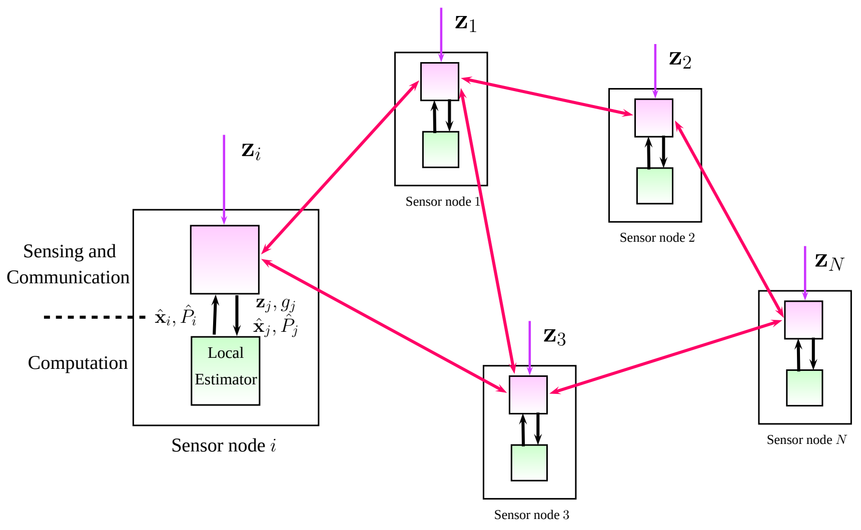


Figure 2.1: *Proposed distributed (sensing) framework for the estimation problem. We note that each of the local estimators has the same input-output structure to allow the possibility to "plug and play" different local filters.*

We recall that in standard centralized approaches, all the measurement set $\{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ would be sent to a centralized estimator, which would deliver an estimate for the state. Instead, in a distributed setting approach there are a number of sensor nodes that locally observe the process and communicate one another to compute a common estimate for the state. In Figure 2.1 the proposed distributes (sensing) approach is illustrated. We divide each of the sensor nodes into two parts, the sensing and communication part and the computation part. The sensing and communication part is responsible for measuring the quantity $\mathbf{z}_i(k)$ and communicating with the neighbors sensor nodes. The messages consist of $\mathbf{z}_i, g_i$, $\hat{\mathbf{x}}_i(x, \tau)$ and its covariance $\hat{P}_i(k, \tau)$ (Where we denote $\hat{\mathbf{x}}_i(k, 0) = \bar{\mathbf{x}}_i(k)$). The sensing and communication part is connected to the computation part that is responsible of estimating the state via a local estimator. This local estimator receives as an input the available variables $\mathbf{z}_j, g_j, \hat{\mathbf{x}}_j(k, \tau - 1), \hat{P}_j(k, \tau - 1)$ (that come from its own and neighboring sensor nodes) and deliver as an output its own value of $\hat{\mathbf{x}}_i(k, \tau), \hat{P}_i(k, \tau)$. This input-output structure is the same across the network and the same for different local estimators.

In this context, we note that the main difference of the proposed structure with the available literature is that each local estimator in Figure 2.1 will be constructed in order to have the same input-output structure, which enables us to "plug and play" different filters and have an heterogeneous group of estimators as a result. Although similar concepts have been applied in centralized settings (Rajamani and Rawlings, 2007, Qu and Hahn, 2009, Ungarala, 2009), this is an important novelty in the distributed domain. In particular, this feature enables to tailor the local filters to the different sensors devices (and therefore hardware) that are available in practice, which is of critical importance in sensor network applications.

The proposed structure is based on a weighted consensus mechanism that merges the local estimates and the covariance matrices coming from the different sensor nodes, as illustrated next.

2.2.4 Weighted Consensus Algorithm

As expressed in the problem formulation we let $\bar{\mathbf{x}}_i(k)$ and $\bar{P}_i(k)$ be the local state estimate of sensor node i and its covariance matrix before agreements with the neighboring nodes. Let $\hat{\mathbf{x}}_i(k,\tau)$ and $\hat{P}_i(k,\tau)$ be the values of the nodal estimate and the covariance matrix after τ rounds of communication with the neighboring nodes. Often, in order to simplify the formalism, with abuse of notation, we will denote

$$\hat{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i(k,\tau) \quad \text{and} \quad \hat{P}_i(k) = \hat{P}_i(k,\tau).$$

We will use averaging consensus algorithms to implement the agreement protocol among the sensor nodes, which will be important, not only to allow the sensor nodes to agree on a common state estimate (Requirement *(iii)* of the Distributed Estimation Problem (2.1)), but also to improve the distributed method's accuracy. Standard references to these types of algorithms are Olfati-Saber and Murray (2004), Olfati-Saber et al. (2007), Cortés (2008), Keviczky and Johansson (2008), Ren and Beard (2008). In particular, we consider recursive merging iterations of the form

$$\begin{aligned} \hat{\mathbf{x}}_i(k,0) &= \bar{\mathbf{x}}_i(k) \quad \text{for all } i \in \mathcal{V}, \\ \hat{\mathbf{x}}_i(k,\kappa) &= \sum_{j\in\mathcal{N}_i^+} w_{ij}\bar{\mathbf{x}}_j(k,\kappa-1), \quad \kappa = 1,\dots,\tau, \end{aligned} \tag{2.8}$$

where $w_{ij} \in \mathbb{R}$. Let $W \in \mathbb{R}^{N\times N}$ be the matrix with entries w_{ij}. We can represent the iterations (2.8) in a matrix-vector form as

$$\begin{pmatrix} \hat{\mathbf{x}}_1(k,\kappa) \\ \hline \hat{\mathbf{x}}_2(k,\kappa) \\ \hline \vdots \\ \hline \hat{\mathbf{x}}_N(k,\kappa) \end{pmatrix} = \underbrace{\left[\begin{array}{c|c|c|c} w_{11}I_n & w_{12}I_n & \dots & w_{1N}I_n \\ \hline w_{21}I_n & w_{22}I_n & \dots & w_{2N}I_n \\ \hline & & & \\ \hline w_{N1}I_n & w_{N2}I_n & \dots & w_{NN}I_n \end{array}\right]}_{W\otimes I_n} \begin{pmatrix} \hat{\mathbf{x}}_1(k,\kappa-1) \\ \hline \hat{\mathbf{x}}_2(k,\kappa-1) \\ \hline \vdots \\ \hline \hat{\mathbf{x}}_N(k,\kappa-1) \end{pmatrix},$$

where $w_{ij} = 0$ if i and j are not neighboring sensor nodes.

As in standard averaging consensus algorithms, we require that the matrix W satisfies (Ren and Beard, 2008)

$$\lim_{\tau\to\infty} W^\tau = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top, \tag{2.9}$$

where $\mathbf{1}_N$ is a vector of dimension N of all ones. With the property (2.9) the consensus iterations (2.8) converge to a final state, where all the local variables are equal to the mean of the initial values. This fact is used to satisfy requirement *(iii)* of the Distributed Estimation Problem 2.1.

In this chapter, we propose to use a special consensus mechanism, based on a weighted version of the standard iterations (2.8), similar to the algorithm presented in (Xiao et al., 2005). In particular, in the standard iterations (2.8) with the property (2.9) the sensor nodes agree on the average value of the local estimates $\bar{\mathbf{x}}_i(k)$, i.e.,

$$\lim_{\tau\to\infty} \hat{\mathbf{x}}(k,\tau) = \frac{1}{N}\sum_{i\in\mathcal{V}} \bar{\mathbf{x}}_i(k).$$

On the contrary, in our proposed algorithm, they reach an agreement on the average of the local estimates $\bar{\mathbf{x}}_i(k)$ weighted on the covariance matrices $\bar{P}_i(k)$, i.e.,

$$\lim_{\tau\to\infty} \hat{\mathbf{x}}(k,\tau) = \left(\sum_{i\in\mathcal{V}} \bar{P}_i^{-1}(k)\right)^{-1} \left(\sum_{i\in\mathcal{V}} \bar{P}_i^{-1}(k)\bar{\mathbf{x}}_i(k)\right).$$

This weighted average gives more emphasis to local estimates with “smaller” covariance matrices, as one would do intuitively, thus one can expect that this average is a better estimate for $\mathbf{x}(k)$. In order to formalize these ideas we cite the following lemma.

Lemma 2.1 *(Xiao et al., 2005) Given a set of independent and unbiased estimates, $\bar{\mathbf{x}}_i$, with associated covariance matrices, $\bar{P}_i$, where $i\in\mathcal{V}$, the following weighted averaging:*

$$\begin{aligned}\hat{\mathbf{x}} &= \left(\sum_{j\in\mathcal{V}} \bar{P}_j^{-1}\right)^{-1} \sum_{j\in\mathcal{V}} \bar{P}_j^{-1}\bar{\mathbf{x}}_j \\ \hat{P}^{-1} &= \sum_{j\in\mathcal{V}} \bar{P}_j^{-1}\end{aligned}$$

gives the minimum-variance unbiased estimate of $\mathbf{x}$.

In order to see how we can leverage on the result of Lemma 2.1 in our proposed consensus mechanism, we introduce some auxiliary variables. Let $\tilde{\mathbf{x}}_i = \bar{P}_i^{-1}\bar{\mathbf{x}}_i$, be the local weighted estimate, let $Y_i = \bar{P}_i^{-1}$ be the inverse of the covariance matrix, usually referred to as the information matrix, let $\tilde{\mathbf{x}} = (1/N)\sum_{j\in\mathcal{V}}\tilde{\mathbf{x}}_j$ be the average of the weighted estimates, and let $Y = (1/N)\sum_{j\in\mathcal{V}} Y_j$ be the average of the information matrices. The weighted averaging given in Lemma 2.1 can be seen as

$$\hat{\mathbf{x}} = Y^{-1}\tilde{\mathbf{x}} \tag{2.10a}$$
$$\hat{P}^{-1} = NY. \tag{2.10b}$$

In this form, noticing that both Y and $\tilde{\mathbf{x}}$ can be computed asymptotically via standard averaging iterations (2.8), the translation of Lemma 2.1 in a consensus protocol appears clearer. In practice, one would run the iterations (2.8) on the local weighted estimates $\tilde{\mathbf{x}}_i$ and information matrices Y for $\tau\to\infty$ and successively evaluate $\hat{\mathbf{x}}$ and $\hat{P}$ via (2.10).

In our case, however, we restrict τ to be finite, in some situations even to be $\tau = 1$. In this context, the agreed estimates $\hat{\mathbf{x}}_i(k,\tau)$, will not be equal to $Y^{-1}\tilde{\mathbf{x}}(k)$ and therefore they will not deliver a minimum-variance estimate of the state $\mathbf{x}(k)$. Furthermore, the local $\bar{\mathbf{x}}_i(k)$ are in general correlated since the sensor nodes are observing the same model. Nonetheless, first, one can expect a better estimation quality with this weighted consensus with respect to standard averaging consensus, and, second, we can guarantee the unbiasedness of the local $\hat{\mathbf{x}}_i(k)$ (even for $\tau = 1$) as follows.

Lemma 2.2 *Given a set of possibly dependent but unbiased estimates, $\bar{\mathbf{x}}_j$, with associated covariance matrices, $\bar{P}_j$, where $j \in \mathcal{N}_i^+$, the weighted average*

$$\begin{aligned}\hat{\mathbf{x}}_i &= \left(\sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\right)^{-1} \sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\hat{\mathbf{x}}_j \\ \hat{P}_i^{-1} &= \sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\end{aligned}$$

will be an unbiased estimate of $\mathbf{x}$.

Proof. The expected value of $\hat{\mathbf{x}}_i$ can be written as

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{x}}_i] &= \mathbb{E}\left[\left(\sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\right)^{-1} \sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\hat{\mathbf{x}}_j\right] \\ &= \left(\sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\right)^{-1} \sum_{j\in\mathcal{N}_i^+} \bar{P}_j^{-1}\,\mathbb{E}[\hat{\mathbf{x}}_j] = \mathbf{x}\end{aligned}$$

from which the claim follows. □

Algorithm 2.1 MERGE$\left(\{\bar{\mathbf{x}}_1(k),\ldots,\bar{\mathbf{x}}_N(k)\},\{\bar{P}_1(k),\ldots,\bar{P}_N(k)\},W,\tau\right)$ algorithm

```
1: Input: $\{\bar{\mathbf{x}}_1(k),\ldots,\bar{\mathbf{x}}_N(k)\},\{\bar{P}_1(k),\ldots,\bar{P}_N(k)\},W,\tau$
2: Compute the auxiliary variables for each $i$: $\tilde{\mathbf{x}}_i(0) = \bar{P}_i^{-1}(k)\bar{\mathbf{x}}_i(k)$, $Y_i(0) = \bar{P}_i^{-1}(k)$
3: Consensus step for each $i$:
4: for $\kappa = 1$ to $\tau$ do
5:     Communicate within $\mathcal{N}_i^+$ the couple $(\tilde{\mathbf{x}}_i(\kappa-1), Y_i(\kappa-1))$
6:     Compute:
           $\begin{cases}\tilde{\mathbf{x}}_i(\kappa) = \sum_{j\in\mathcal{N}_i^+} w_{ij}\tilde{\mathbf{x}}_j(\kappa-1)\\ Y_i(\kappa) = \sum_{j\in\mathcal{N}_i^+} w_{ij}Y_j(\kappa-1)\end{cases}$
7: end for
8: Compute for each $i$: $\hat{\mathbf{x}}_i(k,\tau) = Y_i^{-1}(\tau)\tilde{\mathbf{x}}_i(\tau)$, $\hat{P}_i^{-1}(k,\tau) = Y_i(\tau)$
9: Output: $\{\hat{\mathbf{x}}_1(k),\ldots,\hat{\mathbf{x}}_N(k)\},\{\hat{P}_1(k),\ldots,\hat{P}_N(k)\}$
```

We report in Algorithm 2.1 our proposed algorithm. We denote the resulting weighted consensus algorithm, as the MERGE algorithm.

Algorithm 2.1 will be used in the following sections to merge the different local estimates and their covariances coming from the sensor nodes. We note once more that, although it is not guaranteed to deliver a minimum-variance estimate, numerical simulation studies (in addition to the ones illustrated in (Simonetto et al., 2010a, Simonetto and Keviczky, 2012)) will show improved accuracy in delivering state estimates with respect to standard consensus algorithms (2.8).

2.3 Distributed Nonlinear Estimators

In this section we proposed distributed versions of commonly used nonlinear estimation methods, namely the Moving Horizon Estimators, Particle Filters, and Unscented and Extended Kalman Filters. We will start from the Moving Horizon Estimators, which allows the most general assumptions on the system model and constraints. Then, with some more restricting assumptions on the model and the constraints we will discuss Particle Filters, and Unscented and Extended Kalman Filters .

2.3.1 Moving Horizon Estimators

In its full generality, the Distributed Estimation Problem 2.1 is still an open research problem, due to the current incapability for consensus iterations to handle generic non-convex constraints. In this section we study Moving Horizon Estimators that will require the simplifying assumption that the state constraints (2.2) are convex sets.

Centralized formulation

Moving Horizon Estimation (MHE) is an optimization based state estimation technique which has been developed to include constraints and nonlinearities in the problem formulation extending the popular Kalman Filter approach (Rao, 2000, Rao et al., 2003, Haseltine and Rawlings, 2005, Rawlings and Bakshi, 2006, Kang, 2006, Alessandri et al., 2011). This makes MHE particularly suitable for the (Distributed) Estimation Problem 2.1.

Let $\mathbf{x}^{\text{in}}(0)$ be the estimated initial condition for the estimation problem and let $P^{\text{in}}(0)$ be its covariance. Let the set of all process disturbances from $\kappa = t$ to $\kappa = k$ be denoted by $\{\mathbf{w}(\kappa)\}_t^k$. In the standard Kalman Filter approach, one would weight the process noise $\mathbf{w}(k)$ and the measurement noise $\boldsymbol{\mu}_i(k)$ via a quadratic cost function, as

$$J_{\text{KF}} = \frac{1}{2}\left(\sum_{i=1}^{N} ||\boldsymbol{\mu}_i(k)||^2_{R_i^{-1}} + ||\mathbf{w}(k)||^2_{Q^{-1}}\right) \tag{2.11}$$

where $R_i \succ 0$ and $Q \succ 0$ are positive definite matrices of appropriate dimensions and the notation $||\mathbf{v}||^2_A$, with A a matrix of appropriate dimensions, denotes $\mathbf{v}^\top A\mathbf{v}$. In a similar fashion, the first step of the MHE approach is to consider the *centralized*, full-information, cost function

$$J_k\left(\hat{\mathbf{x}}(0), \{\mathbf{w}(\kappa)\}_{\kappa=0}^{k-1}\right) = \frac{1}{2}\left(\sum_{\kappa=1}^{k}\sum_{i=1}^{N} ||\boldsymbol{\mu}_i(\kappa)||^2_{R_i^{-1}} + \sum_{\kappa=0}^{k-1} ||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \frac{1}{2}\left\|\hat{\mathbf{x}}(0) - \mathbf{x}^{\text{in}}(0)\right\|^2_{P^{\text{in}}(0)^{-1}}, \tag{2.12}$$

In this way, the function J_k can be interpreted as a generalization of the Kalman filter cost function J_{KF} (Eq. (2.11)). The term

$$\frac{1}{2}\left\|\hat{\mathbf{x}}(0) - \mathbf{x}^{\text{in}}(0)\right\|^2_{P^{\text{in}}(0)^{-1}}$$

represents our confidence in the estimate of the initial condition. Exploiting the measurement equation (2.3), we can rewrite the cost function (2.12) highlighting the dependence on the state, process noise, and measurements as

$$J_k\left(\hat{\mathbf{x}}(0), \{\mathbf{w}(\kappa)\}_{\kappa=0}^{k-1}\right) = \frac{1}{2}\left(\sum_{\kappa=1}^{k}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=0}^{k-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \frac{1}{2}\left\|\hat{\mathbf{x}}(0) - \mathbf{x}^{\text{in}}(0)\right\|^2_{P^{\text{in}}(0)^{-1}}. \quad (2.13)$$

We remark that J_k depends only on $\hat{\mathbf{x}}(0)$ and $\{\mathbf{w}(\kappa)\}_{\kappa=0}^{k-1}$ since one can reconstruct the whole state trajectory from $\hat{\mathbf{x}}(0)$ till $\hat{\mathbf{x}}(k)$ via the dynamical model (2.1).

Consider the minimization problem

$$\underset{\hat{\mathbf{x}}(0),\{\mathbf{w}(\kappa)\}_{k=0}^{k-1}}{\textbf{minimize}} \quad J_k\left(\hat{\mathbf{x}}(0), \{\mathbf{w}(\kappa)\}_{\kappa=0}^{k-1}\right) \quad (2.14)$$
$$\textbf{subject to}$$
$$\begin{cases} f(\hat{\mathbf{x}}(\kappa-1), \mathbf{w}(\kappa-1)) = \hat{\mathbf{x}}(\kappa) \in \mathbb{X} & \text{for } \kappa = 1, \ldots, k \\ \mathbf{w}(\kappa) \in \mathbb{W} & \text{for } \kappa = 0, \ldots, k-1 \\ \mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa)) = \boldsymbol{\mu}_i(\kappa) \in \mathbb{M}_i & \text{for all } i \text{ and for } \kappa = 1, \ldots, k \end{cases}$$

which delivers the solution pair $(\hat{\mathbf{x}}(0), \{\mathbf{w}(\kappa)\}_{k=0}^{k-1})$ and let the optimal cost function be J_k^{opt}. The constraints of the minimization problem (2.14) are the representation of the initial constraints (2.2) and (2.4). Via the optimizer of (2.14) we can reconstruct the whole state evolution using the dynamical equation (2.3) and therefore estimate the state $\hat{\mathbf{x}}(k)$ at time step k.

In order to solve the optimization problem (2.14) we need to keep in memory all the measurements from $\kappa = 1$ till $\kappa = k$, and the size of the problem grows in time. These aspects make the solution of (2.14) computationally difficult in practice. The basic strategy of MHE is to define an optimization problem using a moving, but fixed-size estimation window and *approximate* the information outside the window. Consider a fixed moving window $T_w = k - T$ and separate the cost function (2.13) as

$$\begin{aligned} J_k\left(\hat{\mathbf{x}}(0), \{\mathbf{w}(\kappa)\}_{\kappa=0}^{k-1}\right) = & \\ & \frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=k-T}^{k-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \\ & \frac{1}{2}\left(\sum_{\kappa=1}^{k-T}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=0}^{k-T-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \\ & \frac{1}{2}\left\|\hat{\mathbf{x}}(0) - \mathbf{x}^{\text{in}}(0)\right\|^2_{P^{\text{in}}(0)^{-1}}. \end{aligned} \quad (2.15)$$

The terms that refer to a time step before $k - T$ (the ones that need to be approximated) form the part of the cost function usually denoted as *arrival cost* or Z_{k-T}:

$$Z_{k-T} = \frac{1}{2}\left(\sum_{\kappa=1}^{k-T}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=0}^{k-T-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \frac{1}{2}\left\|\hat{\mathbf{x}}(0) - \mathbf{x}^{\text{in}}(0)\right\|^2_{P^{\text{in}}(0)^{-1}} . \qquad (2.16)$$

The strategy of the MHE is to solve the fixed size approximated problem

$$\underset{\hat{\mathbf{x}}(k-T),\{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}}{\textbf{minimize}} \quad \hat{J}_k\left(\hat{\mathbf{x}}(k-T), \{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}\right) \qquad (2.17)$$

$$\textbf{subject to}$$

$$\begin{cases} f(\hat{\mathbf{x}}(\kappa-1), \mathbf{w}(\kappa-1)) = \hat{\mathbf{x}}(\kappa) \in \mathbb{X} & \text{for } \kappa = k-T+1, \ldots, k \\ \mathbf{w}(\kappa) \in \mathbb{W} & \text{for } \kappa = k-T, \ldots, k-1 \\ \mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa)) = \boldsymbol{\mu}_i(\kappa) \in \mathbb{M}_i & \text{for all } i \text{ and for } \kappa = k-T+1, \ldots, k \end{cases}$$

which delivers the solution pairs $(\hat{\mathbf{x}}^{\text{mh}}(k-T), \{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1})$ and whose optimal cost is $\hat{J}_k^{\text{opt}}$. In (2.17) the approximated cost function has the form

$$\hat{J}_k\left(\hat{\mathbf{x}}(k-T), \{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}\right) = \frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=k-T}^{k-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right) + \hat{Z}_{k-T}, \qquad (2.18)$$

whereas the approximated arrival cost can be computed as

$$\hat{Z}_{k-T} = \hat{J}_{k-T}^{\text{opt}} + \frac{1}{2}\left\|\hat{\mathbf{x}}(k-T) - \mathbf{x}^{\text{mh}}(k-T)\right\|^2_{P^{\text{mh}}(k-T)^{-1}} . \qquad (2.19)$$

for a suitable choice of the covariance matrix $P^{\text{mh}}(k-T)$. This choice is important for the stability and convergence of the MHE estimator. Usually, $P^{\text{mh}}(k-T)$ is propagated from $P^{\text{in}}(k-T)$ via the Extended Kalman Filter (Rao, 2000), which guarantees stability and convergence. Another possibility is to choose

$$\hat{Z}_{k-T} = \hat{J}_{k-T}^{\text{opt}},$$

which also guarantees stability and convergence (Rao, 2000). In general, one can *enforce* these properties by choosing a scaled approximation of the arrival cost as

$$\hat{Z}_{k-T}^{\beta} = \hat{J}_{k-T}^{\text{opt}} + \frac{\beta(k-T)}{2}\left\|\hat{\mathbf{x}}(k-T) - \mathbf{x}^{\text{mh}}(k-T)\right\|^2_{P^{\text{mh}}(k-T)^{-1}} , \qquad (2.20)$$

for any finite $P^{\text{mh}}(k-T) \succ 0$ and $\beta(k-T) \in [0, 1]$. The parameter $\beta(k-T)$ can be determined on-line to enforce the stability and convergence properties, as explained in detail in (Rao, 2000, Rao et al., 2003). To our purposes we remark that this determination involves solving the auxiliary optimization problem

$$\underset{\hat{\mathbf{x}}(k-T),\{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}}{\textbf{minimize}} \quad \hat{\Phi}_k\left(\hat{\mathbf{x}}(k-T),\{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}\right) \tag{2.21}$$

$$\textbf{subject to}$$

$$\begin{cases} f(\hat{\mathbf{x}}(\kappa-1),\mathbf{w}(\kappa-1))) = \hat{\mathbf{x}}(\kappa) \in \mathbb{X} & \text{for all } \kappa = k-T+1,\dots,k \\ \mathbf{w}(\kappa) \in \mathbb{W} & \text{for all } \kappa = k-T,\dots,k-1 \\ \mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa)) = \boldsymbol{\mu}_i(\kappa) \in \mathbb{M}_i & \text{for all } i \text{ and for all } \kappa = k-T+1,\dots,k \end{cases}$$

where the cost function is

$$\hat{\Phi}_k\left(\hat{\mathbf{x}}(k-T),\{\mathbf{w}(\kappa)\}_{\kappa=k-T}^{k-1}\right) = \frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k}\sum_{i=1}^{N}||\mathbf{z}_i(\kappa) - g_i(\hat{\mathbf{x}}(\kappa))||^2_{R_i^{-1}} + \sum_{\kappa=k-T}^{k-1}||\mathbf{w}(\kappa)||^2_{Q^{-1}}\right),$$

which does not have any arrival cost. Then, given an arbitrary arrival cost $\hat{Z}_{k-T}(\cdot)$, the procedure determines the scaling factor as

$$\beta(k-T) = \max_{\beta\in[0,1]}\left\{\beta : \beta\left(\hat{Z}_{k-T}(\hat{\mathbf{x}}(k-T)) - \hat{J}^{\text{opt}}_{k-T}\right) + \hat{J}^{\text{opt}}_{k-T} \le U(\hat{\Phi}^{\text{opt}}_{k-T})\right\}$$

where $\hat{\Phi}^{\text{opt}}_{k-T}$ is the optimal value for the cost function $\hat{\Phi}_{k-T}$ of the auxiliary problem and $U(\cdot)$ a specified function[2] of $\hat{\Phi}^{\text{opt}}_{k-T}$.

The MHE idea can thus be summarized as solving the optimization problem (2.17) with a suitable choice of the approximated arrival cost. The optimizer of (2.17) is composed of the state estimate $\hat{\mathbf{x}}^{\text{mh}}(k-T)$ at the beginning of the moving window and the noise sequence $\{\mathbf{w}(k-T),\mathbf{w}(k-T+1),\dots,\mathbf{w}(k-1)\}$. These quantities determine the current state estimate $\hat{\mathbf{x}}(k)$, via the dynamic state equation (2.1), which can be proven to be unbiased (Rao et al., 2003).

The presented traditional centralized problem formulation assumes that all measurements are available in a common location for solving the optimization problem. In the next section, we propose a method to implement the Moving Horizon Estimator in a distributed way using local computational capabilities of the different sensor nodes. The proposed distributed approach is a first step towards the generalization of the work of (Farina et al., 2010) for the case of nonlinear dynamics.

Distributed solution approach

Considering the centralized cost function (2.18), there are two terms for which global information is necessary. One is the measurement term, the other is the arrival cost. Although it is easy to imagine how one would distribute the measurement term by limiting the sharing of measurements to a certain neighborhood, treating the arrival cost in a similar fashion is more difficult to accomplish. In particular, the proofs of stability and convergence of the centralized estimator need to be adapted to the distributed case, which is in

[2] We will give more details on the procedure when explaining the distributed implementation.

general not straightforward. Besides this, an additional requirement would be for each sensor node to eventually converge to the same state estimate value, as expressed in the Distributed Estimation Problem 2.1.

Our approach to handle these issues can be summarized in the following four steps:

1) The exchange of measurements is limited to within the neighborhood of each sensor, i.e., $\mathcal{N}_i^+$.[3]

2) The arrival cost is approximated by implementing the weighted consensus algorithm 2.1 on the different local couples $(\bar{\mathbf{x}}_i^{\text{mh}}(k-T), \bar{P}_i(k-T))$. Each of the $\bar{P}_i(k-T)$ is computed through a local Extended Kalman Filter, as in the centralized case. The consensus results for each sensor nodes will be denoted by $(\hat{\mathbf{x}}_i^{\text{mh}}(k-T), \hat{P}_i(k-T))$ and in order to ensure stability, we introduce the local scaling factor $\beta_i(k-T) \in [0,1]$ (as done in (Rao et al., 2003) in a centralized setting).

3) Local cost functions are constructed for each sensor node, which constitute the core of the "local estimator" part of the algorithm (see Figure 2.1), as

$$\hat{J}_{i,k}\left(\bar{\mathbf{x}}_i(k-T), \{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) =$$
$$\frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k} \sum_{j\in\mathcal{N}_i^+} ||\mathbf{z}_j(\kappa) - g_j(\bar{\mathbf{x}}_i(\kappa))||^2_{R_j^{-1}} + \sum_{\kappa=k-T}^{k-1} ||\bar{\mathbf{w}}_i(\kappa)||^2_{Q^{-1}}\right)$$
$$+ \hat{J}^{\text{opt}}_{i,k-T} + \frac{\beta_i(k-T)}{2}||\bar{\mathbf{x}}_i(k-T) - \hat{\mathbf{x}}_i^{\text{mh}}(k-T)||^2_{\hat{P}_i(k-T)^{-1}}, \tag{2.22}$$

with $R_j \succ 0$, $Q \succ 0$. We note that the local cost function (2.22) is a locally computable version of the centralized cost function in the MHE formulation (2.18). The local MHE-optimization problem (corresponding to the "local estimator" step in Figure 2.1) can thus be summarized as

[3]This exchange is (implicitly) assumed to guarantee observability for the local estimators. We remark that in the context of Moving Horizon Estimators the observability rank condition (Eq. (2.6)) can be relaxed over the considered time window; this concept is known as uniform observability. Formally, a system is uniformly observable if there exists a positive integer n_o and a K-function $\phi(\cdot)$ such that for any two states $\mathbf{x}_1(k)$ and $\mathbf{x}_2(k)$

$$\phi(||\mathbf{x}_1(k) - \mathbf{x}_2(k)||) \le \sum_{\kappa=k}^{k+n_o-1} ||g(\mathbf{x}_1(\kappa)) - g(\mathbf{x}_2(\kappa))||, \quad \text{for all } k \ge 0.$$

Using this relaxed definition, the local exchange of measurements is assumed to satisfy the following relation

$$\phi(||\mathbf{x}_1(k) - \mathbf{x}_2(k)||) \le \sum_{\kappa=k}^{k+n_o-1} \sum_{j\in\mathcal{N}_i^+} ||g_j(\mathbf{x}_1(\kappa)) - g_j(\mathbf{x}_2(\kappa))||, \quad \text{for all } i, \text{and for all } k \ge 0.$$

$$\underset{\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(k)\}_{\kappa=k-T}^{T-1}}{\textbf{minimize}} \quad \hat{J}_{i,k}\left(\bar{\mathbf{x}}_i(k-T), \{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) \tag{2.23}$$

$$\textbf{subject to}$$

$$\begin{cases} f(\bar{\mathbf{x}}_i(\kappa-1), \bar{\mathbf{w}}_i(\kappa-1))) = \bar{\mathbf{x}}_i(\kappa) \in \mathbb{X} & \text{for } \kappa = k-T+1, \ldots, k \\ \bar{\mathbf{w}}_i(\kappa) \in \mathbb{W} & \text{for } \kappa = k-T, \ldots, k-1 \\ \mathbf{z}_j(\kappa) - g_j(\bar{\mathbf{x}}_i(\kappa)) = \boldsymbol{\mu}_j(\kappa) \in \mathbb{M}_j & \text{for all } j \in \mathcal{N}_i^+ \\ & \text{and for } \kappa = k-T+1, \ldots, k \end{cases}$$

which delivers the optimal $\hat{J}_{i,k}^{\text{opt}}$ and the solution pair

$$\bar{\mathbf{x}}_i(k-T), \quad \{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}$$

from which the local state estimate at the current time k, i.e., $\bar{\mathbf{x}}_i(k)$ can be determined via the dynamics (2.1).

4) A standard consensus step (Equation (2.8)) is performed using the local state estimates, $\bar{\mathbf{x}}_i(k)$, in order to agree on $\hat{\mathbf{x}}(k)$. We refer to this step as a posteriori consensus step, which is used to facilitate the convergence to the same estimate by each sensor nodes. Since the state estimate has to be feasible with respect to the state constraint set $\mathbb{X}$ even after the agreement process, we introduce the following simplifying assumption.

Assumption 2.1 *The state constraint set $\mathbb{X}$ in (2.2) is convex.*

The local formulation of the filter differs from the centralized setting in different aspects. First of all the scaling factor $\beta_i(k-T)$ is computed locally. This is done using the same procedure as in (Rao et al., 2003) but localized on each sensor node. Second, the arrival cost is based on agreed values of the couple $(\hat{\mathbf{x}}_i^{\text{mh}}(k-T), \hat{P}_i(k-T))$. Although this seems rather natural, in general the agreed $\hat{\mathbf{x}}_i^{\text{mh}}(k-T)$ may not be in the reachable set of the dynamical system (2.1), which could lead to worse performance for the estimation than the centralized implementation. We note that this effect is due to the nonlinear nature of the problem and it is not present in the linear case with convex constraints (Farina et al., 2010).

Solution properties

Under very general regularity assumptions[4] on the dynamics, measurement equation, and cost function and under particular conditions on the arrival cost, the centralized Moving Horizon Estimator is stable and delivers an unbiased estimate for the state $\mathbf{x}(k)$ (Rao et al., 2003). This is also true for the local estimators if their arrival cost verifies the same conditions of the centralized case, meaning

[4]These conditions require f and g to be Lipschitz, the cost function to be quadratic, and the optimization problem to be well-posed.

C1) There exists a K-function[5] $\bar{\gamma}(\cdot)$ such that

$$0 \leq \hat{Z}_{i,k-T}(z) - \hat{J}^{\text{opt}}_{i,k-T} \leq \bar{\gamma}(||z - \hat{\mathbf{x}}_i^{\text{mh}}(k-T)||) \tag{2.24}$$

C2) The sequence of the arrival costs $\{\hat{Z}_{i,k-T}\}$ is monotonically non-increasing.

Condition **C1** is satisfied with our choice

$$\hat{Z}_{i,k-T}(z) = \hat{Z}^{\beta}_{i,k-T}(z) = \hat{J}^{\text{opt}}_{i,k-T} + \frac{\beta_i(k-T)}{2}||z - \hat{\mathbf{x}}_i^{\text{mh}}(k-T)||^2_{\hat{P}_i(k-T)^{-1}},$$

while condition **C2** can be enforced as explained in (Rao et al., 2003) using the scaling factor $\beta_i(k-T) \in [0,1]$ that forces the sequence of arrival costs to be monotonically non-increasing. This involves solving the local auxiliary optimization problem

$$\underset{\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}}{\textbf{minimize}} \quad \hat{\Phi}_{i,k}\left(\bar{\mathbf{x}}_i(k-T), \{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) \tag{2.25}$$

$$\textbf{subject to}$$

$$\begin{cases} f(\bar{\mathbf{x}}_i(\kappa-1), \bar{\mathbf{w}}_i(\kappa-1))) = \bar{\mathbf{x}}_i(\kappa) \in \mathbb{X} & \text{for } \kappa = k-T+1, \dots, k \\ \bar{\mathbf{w}}_i(\kappa) \in \mathbb{W} & \text{for } \kappa = k-T, \dots, k-1 \\ \mathbf{z}_j(\kappa) - g_j(\bar{\mathbf{x}}_i(\kappa)) = \boldsymbol{\mu}_j(\kappa) \in \mathbb{M}_j & \text{for all } j \in \mathcal{N}_i^+ \\ & \text{ and for } \kappa = k-T+1, \dots, k \end{cases}$$

where the cost function is

$$\hat{\Phi}_{i,k}\left(\bar{\mathbf{x}}_i(k-T), \{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) = \frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k} \sum_{j \in \mathcal{N}_i^+} ||\mathbf{z}_j(\kappa) - g_j(\bar{\mathbf{x}}_i(\kappa))||^2_{R_j^{-1}} + \sum_{\kappa=k-T}^{k-1} ||\bar{\mathbf{w}}_i(\kappa)||^2_{Q^{-1}}\right).$$

We note that the optimization problem (2.25) is the local version of the centralized (2.21). The optimal cost function of (2.25) is $\hat{\Phi}^{\text{opt}}_{i,k}$. Define

$$\tilde{J}^{\text{opt}}_{i,k} = \begin{cases} \hat{J}^{\text{opt}}_{i,k}, & \text{if } k \leq T, \\ \hat{\Phi}^{\text{opt}}_{i,k} + \tilde{J}^{\text{opt}}_{i,k-T}, & \text{if } k > T \end{cases}.$$

The monotonically non-decreasing condition for the arrival cost can be written as (Rao et al., 2003)

$$\hat{Z}_{i,k}(\bar{\mathbf{x}}_i(k)) \leq \min_{\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(k)\}_{\kappa=k-T}^{T-1}} \frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k} \sum_{j \in \mathcal{N}_i^+} ||\mathbf{z}_j(\kappa) - g_j(\bar{\mathbf{x}}_i(\kappa))||^2_{R_j^{-1}} + \sum_{\kappa=k-T}^{k-1} ||\bar{\mathbf{w}}_i(\kappa)||^2_{Q^{-1}}\right) + \tilde{J}^{\text{opt}}_{i,k-T} = \tilde{J}^{\text{opt}}_{i,k}.$$

[5] A function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ is a K-function if it is continuous, strictly monotone increasing, $\alpha(x) > 0$ for $x \neq 0$, $\alpha(0) = 0$, and $\lim_{x\to\infty} \alpha(x) = \infty$.

This leads to the procedure to determine $\beta_i(k-T)$ (locally) and $\hat{Z}^{\beta}_{i,k-T}(\cdot)$:

- pick any $\hat{Z}_{i,k-T}(\cdot)$ satisfying **C1**;
- compute $\beta_i(k-T)$ as
$$\beta_i(k-T) = \max_{\beta\in[0,1]} \left\{\beta : \beta\left(\hat{Z}_{i,k-T}(\bar{\mathbf{x}}_i(k-T)) - \hat{J}^{\text{opt}}_{i,k-T}\right) + \hat{J}^{\text{opt}}_{i,k-T} \le \tilde{J}^{\text{opt}}_{i,k-T}\right\};$$
- set
$$\hat{Z}^{\beta}_{i,k-T}(\cdot) = \beta_i(k-T)\left(\hat{Z}_{i,k-T}(\bar{\mathbf{x}}_i(k-T)) - \hat{J}^{\text{opt}}_{i,k-T}\right) + \hat{J}^{\text{opt}}_{i,k-T}.$$

Although under condition **C1** and **C2**, we can prove stability and unbiasedness of the local estimates, we underline that our choice of merging mechanism (necessary to improve the estimation quality of the distributed implementation with respect to non-communicating local filters) could worsen the performance of the distributed estimator with respect to a centralized implementation. In order to understand better the nature of this problem, consider the local couple $(\bar{\mathbf{x}}^{\text{mh}}_i(k-T), \bar{\mathbf{w}}_i(k-T))$ and the agreed $\hat{\mathbf{x}}^{\text{mh}}_i(k-T)$. By the use of the nonlinear dynamical equation (2.1) we impose that

$$\bar{\mathbf{x}}^{\text{mh}}_i(k-T+1) = f(\bar{\mathbf{x}}^{\text{mh}}_i(k-T), \bar{\mathbf{w}}_i(k-T)), \quad \text{for all } i.$$

However, after the agreement process (necessary to incorporate the neighbors information into the estimator), it may happen that no vector $w \in \mathbb{W}$ can satisfy

$$\hat{\mathbf{x}}^{\text{mh}}_i(k-T+1) = f(\hat{\mathbf{x}}^{\text{mh}}_i(k-T+1), w), \quad \text{for all } i,$$

meaning that $\hat{\mathbf{x}}^{\text{mh}}_i(k-T)$ is not reachable. This translates in the fact that in the following step of the local MHE problem ($k \leftarrow k+1$), the term

$$||\bar{\mathbf{x}}_i(k-T) - \hat{\mathbf{x}}^{\text{mh}}_i(k-T)||^2_{\hat{P}_i(k-T)^{-1}}$$

drives the local estimate $\bar{\mathbf{x}}_i(k-T)$ to the non-reachable set. The detailed study of this phenomenon is left as future research direction.

Algorithm 2.2 summarizes our proposed distributed estimation strategy taking into consideration all the aspects discussed above.

We conclude this section considering once more the requirements of the Distributed Estimation Problem 2.1 and some remaining challenges. We note that by construction the estimate $\hat{\mathbf{x}}_i(k)$ satisfies the constraints (req. *(i)*), while its unbiasedness (req. *(ii)*) holds. Finally, requirement *(iii)* is enforced via the presence of a posteriori consensus which brings the different state estimates to converge to the same value (when $\tau \to \infty$).

We will analyze the performance of Algorithm 2.2 in numerical simulation in Section 2.4, while in the following we will remove the constraints from the formulation of the Distributed Estimation Problem 2.1 and explore other possible estimators.

Remark 2.2 (Centralized and distributed algorithms' performance) *In general, due to the nonlinear nature of the optimization problem and measurement data sharing, the distributed estimator will*

Algorithm 2.2 Distributed MHE

```
1: Input: $\{\hat{\mathbf{x}}_1^{\mathrm{mh}}(k-T),\dots,\hat{\mathbf{x}}_N^{\mathrm{mh}}(k-T)\},\{\hat{P}_1(k-T),\dots,\hat{P}_N(k-T)\},\{\mathbf{z}_1(k),\dots,\mathbf{z}_N(k)\}$
   $\triangleright$ Available Data: $f, g_i, \mathbb{X}, \mathbb{W}, \mathbb{M}_i, R_i, Q, \tau_1, \tau_2$
2: Sharing: each sensor node shares $(\mathbf{z}_i(k), g_i, \mathbb{M}_i)$ with its neighbors to achieve local observability
3: Local Estimation for each $i$:
   3.1: Construct local auxiliary cost function as
        $\hat{\Phi}_{i,k}\left(\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) =$
        $\frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k}\sum_{j\in\mathcal{N}_i^+}||\mathbf{z}_j(\kappa)-g_j(\bar{\mathbf{x}}_i(\kappa))||^2_{R_j^{-1}}+\sum_{\kappa=k-T}^{k-1}||\bar{\mathbf{w}}_i(\kappa)||^2_{Q^{-1}}\right)$
   3.2: Solve the minimization
        $\underset{\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}}{\mathbf{minimize}}\ \hat{\Phi}_{i,k}\left(\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right)$
        subject to
        $f(\bar{\mathbf{x}}_i(\kappa-1),\bar{\mathbf{w}}_i(\kappa-1))) = \bar{\mathbf{x}}_i(\kappa)\in\mathbb{X}$   for $\kappa = k-T+1,\dots,k$
        $\bar{\mathbf{w}}_i(\kappa)\in\mathbb{W}$   for $\kappa = k-T,\dots,k-1$
        $\mathbf{z}_j(\kappa)-g_j(\bar{\mathbf{x}}_i(\kappa)) = \boldsymbol{\mu}_j(\kappa)\in\mathbb{M}_j$   for all $j\in\mathcal{N}_i^+$
                                                      and for $\kappa = k-T+1,\dots,k$
   3.3: Determine $\beta_i(k-T)$ as in (Rao et al., 2003)
   3.4: Construct a local cost function as
        $\hat{J}_{i,k}\left(\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right) =$
        $\frac{1}{2}\left(\sum_{\kappa=k-T+1}^{k}\sum_{j\in\mathcal{N}_i^+}||\mathbf{z}_j(\kappa)-g_j(\bar{\mathbf{x}}_i(\kappa))||^2_{R_j^{-1}}+\sum_{\kappa=k-T}^{k-1}||\bar{\mathbf{w}}_i(\kappa)||^2_{Q^{-1}}\right)$
        $+\hat{J}_{i,k-T}^{\mathrm{opt}}+\frac{\beta_i(k-T)}{2}||\bar{\mathbf{x}}_i(k-T)-\hat{\mathbf{x}}_i^{\mathrm{mh}}(k-T)||^2_{\hat{P}_i(k-T)^{-1}}$
   3.5: Solve the minimization
        $\underset{\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(k)\}_{\kappa=k-T}^{T-1}}{\mathbf{minimize}}\ J_{i,k}\left(\bar{\mathbf{x}}_i(k-T),\{\bar{\mathbf{w}}_i(\kappa)\}_{\kappa=k-T}^{k-1}\right)$
        subject to
        $f(\bar{\mathbf{x}}_i(\kappa-1),\bar{\mathbf{w}}_i(\kappa-1))) = \bar{\mathbf{x}}_i(\kappa)\in\mathbb{X}$   for $\kappa = k-T+1,\dots,k$
        $\bar{\mathbf{w}}_i(\kappa)\in\mathbb{W}$   for $\kappa = k-T,\dots,k-1$
        $\mathbf{z}_j(\kappa)-g_j(\bar{\mathbf{x}}_i(\kappa)) = \boldsymbol{\mu}_j(\kappa)\in\mathbb{M}_j$   for all $j\in\mathcal{N}_i^+$
                                                      and for $\kappa = k-T+1,\dots,k$
   3.6: Determine $\bar{P}_i(k-T+1)$ via an Extended Kalman Filter update as in (Rao et al., 2003)
   3.7: Determine the state estimate $\bar{\mathbf{x}}_i(T)$, via the dynamic state equation (2.1)
4: Sharing/Consensus:
   4.1: Consensus on the arrival cost for $\tau_1$ iterations
        $\left(\{\hat{\mathbf{x}}_1(k-T+1),\dots,\hat{\mathbf{x}}_N(k-T+1)\},\{\hat{P}_1(k-T+1),\dots,\hat{P}_N(k-T+1)\}\right) =$
        $\textsc{Merge}\left(\{\bar{\mathbf{x}}_1(k-T+1),\dots,\bar{\mathbf{x}}_N(k-T+1)\},\{\bar{P}_1(k-T+1),\dots,\bar{P}_N(k-T+1)\},W,\tau_1\right)$
        $\hat{\mathbf{x}}_i^{\mathrm{mh}}(k-T+1) = \hat{\mathbf{x}}_i(k-T+1)$ for each $i$
   4.2 A posteriori consensus on the local state estimates at $k$ using (2.8) for $\tau_2$ iterations
        $\hat{\mathbf{x}}_i^{\mathrm{mh}}(k) = \hat{\mathbf{x}}_i(k)$ for each $i$
5: Output: $\{\hat{\mathbf{x}}_1^{\mathrm{mh}}(k),\dots,\hat{\mathbf{x}}_N^{\mathrm{mh}}(k)\}$,
   $\{\hat{\mathbf{x}}_1^{\mathrm{mh}}(k-T+1),\dots,\hat{\mathbf{x}}_N^{\mathrm{mh}}(k-T+1)\},\{\hat{P}_1(k-T+1),\dots,\hat{P}_N(k-T+1)\}$
```

not have the same performance of the centralized one, not even in the asymptotic sense. In particular the difference between the distributed and centralized estimate, i.e., $||\hat{\mathbf{x}}_i(k) - \hat{\mathbf{x}}(k)||$, *will be always strictly positive,* $||\hat{\mathbf{x}}_i(k) - \hat{\mathbf{x}}(k)|| > 0$, *even when* $\tau_1, \tau_2 \to \infty$. *Nonetheless, we remark once more that in many scenarios, the number of sensor nodes could be simply too high to run the centralized estimator and distributed solutions are required.*

2.3.2 Particle Filters

The first estimators that we study for the case of no constraints are Particle Filters.

Centralized formulation

Although Particle Filters have a long research record since their first appearance (Gordon et al., 1993), they still represent an active area of investigation. Due to their generality and simplicity, they have become a topic of constantly growing interest, development, and numerous applications.

We start with the following simplifying assumption.

Assumption 2.2 *The constraint sets in* (2.2) *and* (2.4) *are* $\mathbb{X} = \mathbb{R}^n$, $\mathbb{W} = \mathbb{R}^{\mathrm{w}}$, *and* $\mathbb{M}_i = \mathbb{R}^{\mathrm{q}_i}$.

Furthermore let the process noise be modeled by the probability density function, or PDF, $\pi_{\mathbf{w}}(\mathbf{w})$, and let $\pi_{\boldsymbol{\mu}_i}(\boldsymbol{\mu}_i)$ be the PDF that models the measurement noise.

Particle Filters estimate the state $\mathbf{x}(k)$ via the a posteriori conditional PDF $p(\mathbf{x}(k)|\mathbf{z}(k))$. Since, in most cases, this a posteriori PDF cannot be evaluated because of the complexity of the underlying dynamical system (2.1), the basic idea is to draw m random samples, or particles, $\{\mathbf{x}(k)^j\}_{j=1,\dots,m}$ from a given proposal distribution $q(\mathbf{x}(k)|\mathbf{z}(k))$ with the same support as $p(\mathbf{x}(k)|\mathbf{z}(k))$. Often this proposal distribution is chosen to be the a priori distribution $p(\mathbf{x}(k)|\mathbf{x}(k-1))$, as done in Sample Importance Resample (or SIR) filters. Adopting this choice, the random samples can be computed recursively as

$$\begin{aligned} \mathbf{x}(0)^j &= \mathbf{x}(0) \qquad \text{for } j = 1, \dots, m \\ \mathbf{x}(k)^j &= f(\mathbf{x}(k-1)^j, \mathbf{w}(k-1)^j), \text{ for } k = 1, 2, \dots \end{aligned} \tag{2.26}$$

where $\mathbf{x}(k-1)^j$ is the j-th sample at the discrete time $k-1$ and $\mathbf{w}(k-1)^j$ is randomly drawn from the process noise PDF, i.e., $\mathbf{w}(k-1)^j \sim \pi_{\mathbf{w}}(\mathbf{w})$. To these samples are then associated weights $w(k)^j$ that quantify the likelihood of the sample given the measurements $\mathbf{z}(k)$. The weights are also computed recursively via (Arulampalam et al., 2002)

$$\begin{aligned} w(0)^j &= 1/m \qquad \text{for } j = 1, \dots, m \\ w(k)^j &= \frac{p(\mathbf{x}(k)^j|\mathbf{z}(k))}{q(\mathbf{x}(k)^j|\mathbf{z}(k))} = \frac{p(\mathbf{x}(k)^j|\mathbf{z}(k))}{p(\mathbf{x}(k)^j|\mathbf{x}(k-1)^j)} = \\ &= w(k-1)^j p(\mathbf{z}(k))|\mathbf{x}(k)^j), \text{ for } k = 1, 2, \dots \end{aligned} \tag{2.27}$$

where $p(\mathbf{z}(k))|\mathbf{x}(k)^j)$ is the likelihood that the measurerement $\mathbf{z}(k)$ is observed given the sample $\mathbf{x}(k)^j$. If the measurement noise PDF $\pi_{\boldsymbol{\mu}_i}(\boldsymbol{\mu}_i)$ are Gaussian with zero mean and Σ_i as covariance matrix, then (2.27) can be simplified (up to a normalization) into

$$w(k)^j = w(k-1)^j \exp\left(- \sum_{i=1}^{N} \left\| \mathbf{z}_i(k) - g_i(\mathbf{x}(k)^j) \right\|^2_{\Sigma_i^{-1}} \right) \quad \text{for } j = 1, \dots, m. \tag{2.28}$$

Given the weighted couples $(\mathbf{x}(k)^j, w(k)^j)$, we can approximate the a posteriori PDF $p(\mathbf{x}(k)|\mathbf{z}(k))$ by the use of Dirac's deltas, δ, as

$$p(\mathbf{x}(k)|\mathbf{z}(k)) \approx \hat{p}(\mathbf{x}(k)|\mathbf{z}(k)) = \frac{1}{\Omega(k)} \sum_{j=1}^{m} w(k)^j \delta(\mathbf{x}(k) - \mathbf{x}(k)^j) \tag{2.29}$$

with $\Omega(k) = \sum_j w(k)^j$. Finally, this approximation of the a posteriori PDF gives means to estimate $\mathbf{x}(k)$ as $\hat{\mathbf{x}}(k)$, which was our objective. For example, we can choose $\hat{\mathbf{x}}(k)$ to be the particle with highest weight.

Often, in addition to sampling and weighting the particles, the particle population is resampled. In this resampling step the particles are redrawn from the approximated discrete a posteriori PDF (2.29). Furthermore, since these new particles are i.i.d. samples coming from (2.29), their weights are set to be identical.

The resampling step is a crucial component of particle filter algorithms. Resampling is necessary since it can provide the chance for "good" particles to be considered with higher probability and produce better and more accurate results. Moreover, it overcomes the degeneracy phenomenon, where after a few iterations, all but one particle will have negligible weights. However, it introduces also other practical issues that need careful attention. First, it limits the opportunity to parallelize since all the particles must be combined. Second, the particles that have high weights are statistically selected many times. This may lead to a loss of diversity among the particles as the resultant samples contain many repeated points. Therefore the choice of the number of particles and of the resampling procedure fundamentally determine the properties of the Particle Filter.

We can summarize a prototypical SIR particle filter algorithm as follows (Arulampalam et al., 2002).

```
1: Draw $m$ samples $\mathbf{x}(k)^j$ from a the a priori distribution $p(\mathbf{x}(k)|\mathbf{x}(k-1))$, using (2.26).
2: Compute the importance weight $w(k)^j$ for each $j$, using (2.27).
3: Compute the state estimate according to the approximated a posteriori PDF (2.29).
4: Resample the set of particles according to the approximated a posteriori PDF (2.29).
5: Set $w(k)^j = 1/m$ for all particles $j$.
```

Distributed Particle Filters and Related Work

With the growth of computational power and the exploitation of parallel architectures, Particles Filters are increasingly being considered as suitable candidates for implementations in parallel or distributed computing architectures. For this reason, the use of the terminology “distributed Particle Filters” could be misinterpreted. In this chapter we refer to distributed Particle Filters to indicate estimators capable of solving the Distributed Estimation Problem 2.1. In Chapter 3 we will introduce the distributed computation Particle Filters as the ones that have access to all the measurement data but the computations are performed in a distributed way over a cluster of computing units.

Although references to distributed Particle Filters date back to the work of (Rosencrantz et al., 2003) and (Coates, 2004), detailed analysis and evaluation studies on their properties have been initiated only recently. The main reason is that in order to combine the solutions of the different local estimators there is a need for coherent particle-weight combinations which are not trivial to obtain without demanding communication. There are three main solutions to this problem: either (1) we select locally some representative particle-weight combination to send to the other sensor nodes, (2) we impose that all the sensor nodes have the same particle population, thus we can send only the weights, or (3) we parametrize the a posteriori distribution with some low dimensional representation and we send the parameters.

The first strategy is presented in (Rosencrantz et al., 2003, Lee and West, 2009). The work of (Rosencrantz et al., 2003) is particularly suited for situations in which the sensor nodes have enough data to run accurate Particle Filters on their own and they need extra information only in some special cases. A typical application is localization in a building: when a sensor node has a clear view of the object to be localized, it can run its own Particle Filter with no extra information. On the contrary, when the object is hidden behind a wall, it needs some data from other sensor nodes that can see the object. The main idea is that each sensor node keeps in memory the entire time-evolution of its particles and all the measurements, and it sends some of the particles to the neighbors. The neighbors decide whether some measurements, at some time instant, are valuable for the senders and they reply with the data.

The approach of (Lee and West, 2009) follows the same philosophy, but uses a different sending strategy. It allows sensor nodes to communicate the particle and weight combinations via a random walk approach, sending them randomly across the network. The authors show that, although less efficient for low dimensional problems (with respect to other methods), this algorithm scales linearly with the number of dimensions and it could be a viable alternative for large dimensional scenarios. Furthermore, using the tools of (Doucet et al., 2001), the authors show that the distributed algorithm converges weakly to a centralized approach when the sensor nodes are allowed to exchange information arbitrary times within each sampling time.

The second strategy for the combination problem is to consider all the sensors to have the exact same particle population. This can be enforced by synchronizing the seed of the random number generators. The works in (Coates, 2004, Farahmand et al., 2011) are based on this assumption. Since all the particles across the network are the same, the basic idea is that the local weights can be combined using

$$\hat{w}(k)^j = \left(\prod_{i=1}^{N} \bar{w}_j(k)^j \right)^{1/N}, \tag{2.30}$$

which can be expressed as a sum, and therefore a consensus iteration, using the log operator. Furthermore, to increase performance, as proposed in (Farahmand et al., 2011) the particles can be drawn from an a priori PDF scaled via a distributed adaptation mechanism that pre-filters it and makes it closer to the a posteriori PDF. The final algorithm in (Farahmand et al., 2011) can be sketched as:

1: Sharing/(Consensus): determine local adapted a priori PDF with a consensus step.
2: Local estimator: local particle filter based on the local adapted a priori PDF.
3: Sharing/Consensus: share the weights and agree on them via a consensus step based on Equation (2.30).

The third strategy to combine the estimates in distributed particle filters is to guarantee that all the sensor nodes have the same representation of the proposal distribution $q(\mathbf{x}(k)|\mathbf{z}(k))$. This idea is exploited in the papers (Sheng et al., 2005, Sheng and Hu, 2005, Gu, 2007, Gu et al., 2008, Gu and Hu, 2009, Liu et al., 2009, Oreshkin and Coates, 2010) where the authors use different parametric models. In particular, the most commonly used representation is the Gaussian Mixture Model, or GMM, which can be written as:

$$q(\mathbf{x}(k)|\mathbf{z}(k)) = \sum_{c=1}^{C} \lambda_c(k) \aleph(\sigma_c(k), \Sigma_c(k)),$$

where C, λ_c, σ_c, and Σ_c are parameters of the model and they represent the chosen number of Gaussians, their relative importance, their mean, and their covariance respectively. We recall that $\aleph(a, B)$ is used to denote a Gaussian with mean a and covariance B. This representation has the drawbacks that, first, the sensor nodes have to agree upon several variables if $C \gg 1$, and second, the local representation is built via an iterative optimization scheme, which requires time and may lead to local minima (see (Sheng et al., 2005) for further details). On the other hand, it is rather easy to generate the parameter set $(C, \lambda_c, \sigma_c, \Sigma_c)$ based on the sample description of the a posteriori distribution, and it is also rather straightforward to propagate the model one step ahead via Kalman Filters.

Proposed Distributed Approach

In this chapter, in order to give the same input-output structure to the local estimators, we propose the use the third strategy outlined above and we select the estimated mean and covariance to represent the proposal distribution $q(\mathbf{x}(k)|\mathbf{z}(k))$ as (Gu et al., 2008, Gu and Hu, 2009). Let

$$\bar{\mathbf{x}}_i(k) = \frac{1}{\Omega(k)} \sum_{j=1}^{m} w_i(k)^j \mathbf{x}_i(k)^j \tag{2.31}$$

$$\bar{P}_i(k) = \frac{1}{\Omega(k)} \sum_{j=1}^{m} w_i(k)^j (\mathbf{x}_i(k)^j - \bar{\mathbf{x}}_i(k))(\mathbf{x}_i(k)^j - \bar{\mathbf{x}}_i(k))^\top \tag{2.32}$$

be the estimated mean and covariance of the particle population after the resampling stage for filter i. As usual, the couple $(\hat{\mathbf{x}}_i(k), \hat{P}_i(k))$ denotes the agreed couple after a merging mechanism. This agreed couple $(\hat{\mathbf{x}}_i(k), \hat{P}_i(k))$ can be propagated one step ahead via the use of the standard prediction step of the Unscented Kalman Filter as in (Julier and Uhlmann, 2004). This generates the new couple $(\hat{\mathbf{x}}_{i,k+1|k}(k), \hat{P}_{i,k+1|k}(k))$. Let

$$q_i(\mathbf{x}(k+1)|\mathbf{z}(k+1)) = \aleph(\hat{\mathbf{x}}_{i,k+1|k}(k), \hat{P}_{i,k+1|k}(k))$$

be the agreed proposal distribution for each sensor node i. Our proposed method can be summarized as done in Algorithm 2.3.

We remark that our Distributed Particle Filter algorithm meets the requirements *(i) - (ii)* of Problem 2.1 on constraint satisfaction and unbiasedness of the estimate (in fact, there are no constraints), while requirement *(iii)* is enforced via the consensus step (see step 4 of Algorithm 2.3).

From Algorithm 2.3 it is clear that the required computations increase with the number of particles used, as well as the accuracy of the filter. In other words, to obtain more accurate results more particles are needed and thus more computation. In particular, even for a low number of states the number of particles could be prohibitive for the capabilities of simple sensor nodes. In the next section we discuss extensions of the Kalman Filter which can provide "faster" solutions under certain specific assumptions.

Algorithm 2.3 Distributed (parametric) Particle Filter

```
1: Input: $\{\hat{\mathbf{x}}_1(k-1), \dots, \hat{\mathbf{x}}_N(k-1)\}$, $\{\hat{P}_1(k-1), \dots, \hat{P}_N(k-1)\}$, $\{\mathbf{z}_1(k), \dots, \mathbf{z}_N(k)\}$
   ▷ Available Data: $f, g_i, \tau$
2: Sharing: each sensor node shares $(\mathbf{z}_i(k), g_i)$ with its neighbors to achieve local observability
3: Local Estimator for each $i$:
   3.1: Propagate the parameters $(\hat{\mathbf{x}}_i(k-1), \hat{P}_i(k-1))$ to $(\hat{\mathbf{x}}_i(k|k-1), \hat{P}_i(k|k-1))$ via an Unscented Kalman Filter as in (Julier and Uhlmann, 2004).
   3.2: Draw samples from $\hat{q}_i(\mathbf{x}(k)|\mathbf{z}(k)) = \aleph(\hat{\mathbf{x}}_i(k|k-1), \hat{P}_i(k|k-1))$
   3.3: Calculate the local weights of the samples
        $w(k)^j = p(\mathbf{z}_\ell(k))|\mathbf{x}(k)^j)$, with $\ell \in \mathcal{N}_i^+$
   3.4: Calculate the local state estimate $\bar{\mathbf{x}}_i(k)$ and the parameters $\bar{P}_i(k)$ based on the a posteriori PDF
        $\bar{\mathbf{x}}_i(k) = \frac{1}{\Omega(k)} \sum_{j=1}^{m} w_i(k)^j \mathbf{x}_i(k)^j$
        $\bar{P}_i(k) = \frac{1}{\Omega(k)} \sum_{j=1}^{m} w_i(k)^j (\mathbf{x}_i(k)^j - \bar{\mathbf{x}}_i(k))(\mathbf{x}_i(k)^j - \bar{\mathbf{x}}_i(k))^\top$
   3.4: Resample using the local weights.
4: Sharing/Consensus:
   4.1: Consensus on the local state estimate and covariance:
        $\left(\{\hat{\mathbf{x}}_1(k), \dots, \hat{\mathbf{x}}_N(k)\}, \{\hat{P}_1(k), \dots, \hat{P}_N(k)\}\right) =$
              $\textsc{Merge}\left(\{\bar{\mathbf{x}}_1(k), \dots, \bar{\mathbf{x}}_N(k)\}, \{\bar{P}_1(k), \dots, \bar{P}_N(k)\}, W, \tau\right)$
5: Output: $\{\hat{\mathbf{x}}_1(k), \dots, \hat{\mathbf{x}}_N(k)\}$, $\{\hat{P}_1(k), \dots, \hat{P}_N(k)\}$
```

2.3.3 Extended and Unscented Kalman Filters

In this section we investigate Extended and Unscented Kalman Filters, which involve different linearizations of the nonlinear dynamical equation (2.1) and of the nonlinear measurement equations (2.3). We start defining an almost linear system as follows.

Definition 2.1 *A system Σ comprised of a dynamical equation function f and a measurement equation function g is almost linear (Bar-Shalom et al., 2001) if*

1. *f and g map Gaussian distributed random inputs to mono-modal Gaussian-like output distribution, with approximately zero mean and zero third order moment;*
2. *when f and g are linearized in the point $\mathbf{x}$, the linearized observability matrix has the same rank as the nonlinear observability map at $\mathbf{x}$.*

We consider the following assumptions.

Assumption 2.3 *The noise terms $\mathbf{w}$ and $\boldsymbol{\mu}_i$ are Gaussian random inputs with zero mean.*

Assumption 2.4 *The nonlinear dynamical equation f and the nonlinear measurement equations g_i constitute an almost linear system.*

Under Assumptions 2.2, 2.3, and 2.4, Extended and Unscented Kalman filters can be used to solve the Distributed Estimation Problem 2.1. The reader is referred to (Bar-Shalom et al., 2001, Julier and Uhlmann, 2004) for the centralized setting.

Remark 2.3 *Note that very often Assumption 2.4 is difficult to verify. Nonetheless, this has not limited the application of Extended and Unscented Kalman filters, which are employed successfully in many scenarios. In practice, one verifies the applicability of these linearized approach a posteriori, i.e., after careful simulation studies.*

For the distributed scenario, we propose in Algorithm 2.4 an extension of the ideas in (Olfati-Saber, 2007a, Simonetto et al., 2010a, Cattivelli and Sayed, 2010) that is suitable to be used in our common framework presented in Figure 2.1. (Note that only the case of the Unscented Kalman Filter is considered in Algorithm 2.4, but the Extended case is similar with local Extended Kalman Filters instead of Unscented ones).

We note that Extended and Unscented Kalman Filters are computationally cheaper than Moving Horizon Estimators and Particle Filters but the stability of the estimator can be an issue when the nonlinearities become important. Generally, the (distributed) Unscented Kalman Filter performs better than the Extended one, typically, at the price of being slightly more computationally expensive. The requirements *(ii)*-*(iii)* of Problem 2.1 on unbiasedness and convergence of the estimate to a common value can be guaranteed. We note that requirement *(i)* here does not apply due to the absence of constraints.

Algorithm 2.4 Distributed Unscented Kalman Filter

```
1: Input: $\{\hat{\mathbf{x}}_1(k-1),\ldots,\hat{\mathbf{x}}_N(k-1)\}$, $\{\hat{P}_1(k-1),\ldots,\hat{P}_N(k-1)\}$, $\{\mathbf{z}_1(k),\ldots,\mathbf{z}_N(k)\}$
   ▷ Available Data: $f, g_i, \tau$
2: Sharing: each sensor node shares $(\mathbf{z}_i(k), g_i)$ with its neighbors to achieve local observability
3: Local Filter:
   Local Unscented Kalman filter with input $\hat{\mathbf{x}}_i(k-1)$, $\hat{P}_i(k-1)$, $\mathbf{z}_i(k)$ and output $\bar{\mathbf{x}}_i(k)$, $\bar{P}_i(k)$
4: Sharing/Consensus:
   Consensus on the estimates
   $\left(\{\hat{\mathbf{x}}_1(k),\ldots,\hat{\mathbf{x}}_N(k)\},\{\hat{P}_1(k),\ldots,\hat{P}_N(k)\}\right) = \textsc{Merge}\left(\{\bar{\mathbf{x}}_1(k),\ldots,\bar{\mathbf{x}}_N(k)\},\{\bar{P}_1(k),\ldots,\bar{P}_N(k)\},W,\tau\right)$
5: Output: $\{\hat{\mathbf{x}}_1(k),\ldots,\hat{\mathbf{x}}_N(k)\}$, $\{\hat{P}_1(k),\ldots,\hat{P}_N(k)\}$
```

2.3.4 Remarks on the Common Framework

In Section 2.3.1, 2.3.2, and 2.3.3 we have studied different distributed estimators. We have highlighted their underlying assumptions and proposed distributed implementations, namely Algorithm 2.2, 2.3, and 2.4. We remark here the similar input-output structure of the algorithms in terms of mean and covariance of the locally computed estimates. This will allow us to decide which local estimator to implement on which sensor node, adapting the computational load to hardware limitations.

We note that, when considering Algorithm 2.2 for the distributed MHE, the sensor nodes need to maintain in memory past values of measurements, estimates, and covariances. This has to be taken in consideration also when we implement an heterogeneous group of local estimators comprised of a number of local MHE. Furthermore, in this case, a value of $\hat{P}_i(k)$ has to be chosen for the local MHE, which is not provided by Algorithm 2.2. A reasonable choice is to let $\hat{P}_i(k)$ be smaller by a defined ratio than the one of the neighboring nodes (that use other types of estimators).

After having analyzed the proposed algorithms in a more abstract fashion, in the following, we show some simulation results to capture their performance in different realistic scenarios.

2.4 Numerical Evaluations and Comparisons

In this section we present two realistic test cases to analyze the proposed algorithms. The first scenario is a localization problem using noisy range measurements for mobile robot tracking, which is representative of the Distributed Robotics Lab under development at the Delft Center for Systems and Control. The second scenario is localization via range-only measurements of an autonomous underwater vehicle, which can represent a scaled model of many existing underwater robotic platforms, see for example (Corke et al., 2007).

In both cases, we define the error $\mathbf{e}_i(k)$ of sensor i at time k, as the distance between the true position $x(k)$ at that time and the one estimated by the sensor node i, i.e.,

$$\mathbf{e}_i(k) = ||\hat{x}_i(k) - x_i(k)||. \tag{2.33}$$

Furthermore, we let the mean error $\mathbf{e}_\mathrm{m}$ be defined as:

$$\mathbf{e}_\mathrm{m} = \frac{1}{NT_\mathrm{f}} \sum_{i=1}^{N} \sum_{k=0}^{T_\mathrm{f}} \mathbf{e}_i(k), \tag{2.34}$$

where T_f is the final time of simulation. Finally we let the average error, $\mathbf{e}_\mathrm{a}$, be the mean error averaged over a number of different simulations T_sim, i.e.,

$$\mathbf{e}_\mathrm{a} = \frac{1}{T_\mathrm{sim}} \sum_{i=1}^{T_\mathrm{sim}} (\mathbf{e}_\mathrm{m})_i \,. \tag{2.35}$$

We remark that this average error $\mathbf{e}_\mathrm{a}$ will be always positive by construction.

In the following, for compactness reasons, we will often use the abbreviations: MHE, PF, UKF, and EKF to indicate Moving Horizon Estimators, Particle Filters, Unscented Kalman Filters, and Extended Kalman Filters, respectively.

2.4.1 The Mobile Robot Simulation Example

The first scenario we use as an illustrative example for evaluation purposes is a distributed mobile robot localization problem using range-only measurements (see Figure 2.2). This is a suitable benchmark since the underlying dynamics is nonlinear, different constraints can be imposed, and the state is unobservable by individual sensors, which justifies the need for communication among them.

We denote the position of a mobile robot on a 2-D space with $x = (x_{(1)}, x_{(2)})^\top$, and let θ be its orientation. The velocity and angular velocity are denoted by v and ω, respectively. Let the state be defined as $\mathbf{x} = (x_{(1)}, x_{(2)}, \theta)^\top$ and the control input be $\mathbf{u} = (v, \omega)^\top$ with additive noise processes denoted by $\mathbf{w} = (\mathbf{w}_v, \mathbf{w}_\omega)^\top$, $\mathbf{w} \sim \aleph(0, Q)$. The nonlinear time-invariant dynamical model of the robot is represented by the following discrete-time unicycle model (Thrun et al., 2005) with sampling time Δt:

$$\begin{aligned}
x(k+1) &= x(k) + \frac{\tilde{v}(k)}{\tilde{\omega}(k)} (\sin(\theta(k) + \tilde{\omega}(k)\Delta t) - \sin\theta(k)) \\
y(k+1) &= y(k) - \frac{\tilde{v}(k)}{\tilde{\omega}(k)} (\cos(\theta(k) + \tilde{\omega}(k)\Delta t) - \cos\theta(k)) \\
\theta(k+1) &= \theta(k) + \tilde{\omega}(k)\Delta t
\end{aligned} \tag{2.36}$$

with

$$\tilde{v}(k) = v(k) + \mathbf{w}_v(k), \quad \tilde{\omega}(k) = \omega(k) + \mathbf{w}_\omega(k).$$

We consider the state to be constrained as $\mathbf{x} \in \mathbb{X}$ which represents the physical space limitations of the robot's movement, with the assumption that the constraint set $\mathbb{X}$ is convex.

We consider N sensors to be placed at a specified height h measuring the ranges to two beacons on the robot. The resulting two (range-only) measurement equations for each

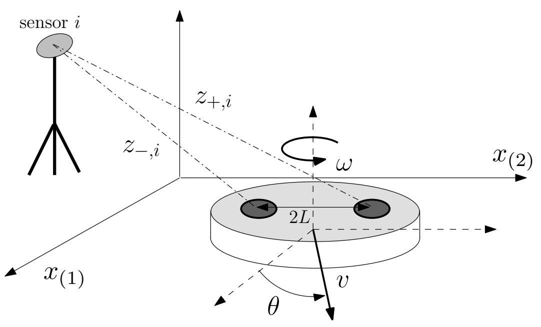


Figure 2.2: *Mobile robot setup description.*

sensor $\mathbf{z}_{\pm,i}(k) = (\mathbf{z}_{+,i}(k), \mathbf{z}_{-,i}(k))^\top$ are given by

$$\mathbf{z}_{\pm,i}(k) \quad = \quad \left\| \begin{Bmatrix} x_{(1)}(k) \pm L\cos\theta(k) - \ell_{i(1)} \\ x_{(2)}(k) \pm L\sin\theta(k) - \ell_{i(2)} \\ h \end{Bmatrix} \right\|_2 + \boldsymbol{\mu}_i(k), \tag{2.37}$$

where $(\ell_{i(1)}, \ell_{i(2)}, h)$ is the position of the sensor, $2L$ is the distance between the beacons and the norm is the standard Euclidean distance. The noise process $\boldsymbol{\mu}_i \sim \aleph(0, R_i)$ is assumed to be Gaussian, which is a common choice when using radio-frequency or ultrasonic ranging devices (Smith et al., 2004). Unless otherwise stated, we will consider the mean of the initial state and its covariance $(\hat{\mathbf{x}}(0), \hat{P}(0))$ as given, or previously estimated.

2.4.2 Simulations with Distributed MHE

Given the presence of constraints, in this test case we use the proposed distributed Moving Horizon Estimator, described by Algorithm 2.2. Moreover, we note that, since there are no constraints on the process noise, no reachability problem will be present.

For the simulation experiments, the sensors are placed and connected as shown in Figure 2.3. The lines represent possible communication links among the sensors, which are marked by squares. The sensors are placed at a height h of 1.5 m. We consider tracking a robot as it traverses through a randomly generated trajectory, shown in Figure 2.4. The simulation parameters are $\Delta t = 1$ s, final time $T_\mathrm{f} = 50$ s, $w_v \sim \aleph(0, (0.01 \text{ m/s})^2)$, $w_\omega \sim \aleph(0, (0.1 \text{ rad/s})^2)$, $\boldsymbol{\mu}_i \sim \aleph(0, (0.05 \text{ m})^2)$, and $P(0) = 0.01I$ with dimensions [m], [m], and [rad], respectively.

The constraints on the position state are represented in Figure 2.4 via shaded areas.

We use Algorithm 2.2 with the parameter $\tau_1 = 1$ in the first consensus process (arrival cost) and we vary τ_2 in the set $\{0, 1, 3, 5\}$. The case $\tau_2 = 0$ represents the choice of no a posteriori consensus (which may be applied when the communication overhead needs to be reduced).

We performed 50 simulation runs of the same trajectory and different random noise processes to investigate the behavior of the distributed estimator with respect to the centralized

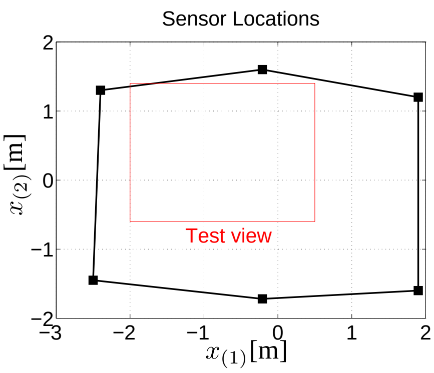


Figure 2.3: *Simulation setup indicating the locations of sensors (squares) and communication links between them (thick lines). The red rectangle represents the constrained experimental area.*

one.

Figure 2.4 depicts the results for a selected representative simulation run using $\tau_2 = 0$. As it can be seen, the distributed MHE solutions of the 6 sensor nodes satisfy the state constraints. Although the communication within the neighborhood is rather limited ($\tau_1 = 1$ and $\tau_2 = 0$), all the local solutions are qualitatively the same.

In Table 2.1 we present the results for the 50 simulations while varying the number of a posteriori consensus iterations τ_2. We can observe that by allowing more a posteriori consensus steps, the estimator delivers better solutions in terms of the average error. We note however that even when $\tau_2 \to \infty$ the distributed solution will not converge to the same solution of the centralized MHE. This is due to the fact that the sensor nodes do not have access to the whole information, since they share measurements and the arrival cost value only with they direct neighbors ($\tau_1 = 1$). Nonetheless, in this limited communication setting, we remark that the error of the distributed estimation is reasonably close to the centralized one given the noise values and communication topology, and therefore the result can be considered completely satisfactory.

Table 2.1: *Average error* $\mathbf{e}_{\mathrm{a}}$ *in the* 50 *simulation runs while varying* τ_2. *CMHE represents the centralized solution.*

	$\tau_2 = 0$	$\tau_2 = 1$	$\tau_2 = 3$	$\tau_2 = 5$	CMHE
Average error $\mathbf{e}_{\mathrm{a}}$, Eq. (2.35) [cm]	4.5	3.6	3.2	3.1	2.5

2.4.3 The Autonomous Underwater Vehicle Simulation Example

The second scenario we consider to compare the methods we have devised for the Distributed Estimation Problem 2.1 is the localization via range-only measurements of an autonomous underwater vehicle.

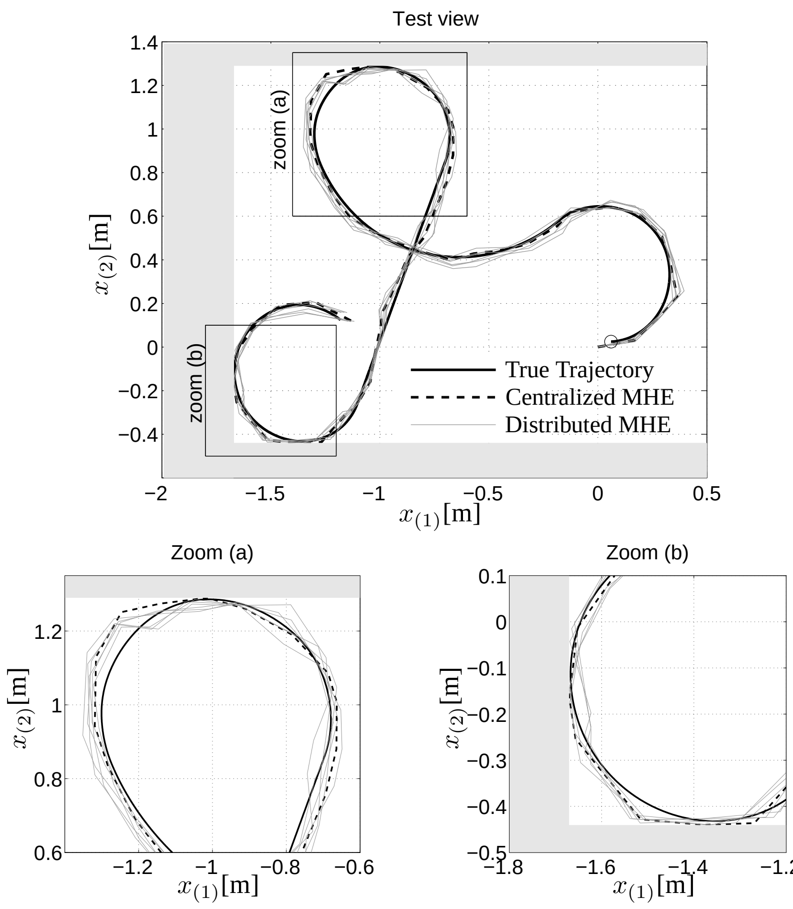


Figure 2.4: *Results of a representative simulation run. The initial position of the robot is marked with a circle, while the shaded areas represent position constraints.*

The state of the AUV is chosen as $\mathbf{x} = (x^\top, v^\top)^\top$, where $x \in \mathbb{R}^3$ is the position and $v \in \mathbb{R}^3$ is the velocity. Let $\mathbf{u}(k) \in \mathbb{R}^3$ be the control input. The discrete-time dynamical equations are:

$$\begin{aligned} x(k+1) &= x(k) + v(k)\Delta t \\ v(k+1) &= v(k) + \frac{\Delta t}{M}\left(\hat{\mathbf{u}}(k) - c_{\mathrm{D}}\,\|v(k)\|\,v(k)\right) \\ \hat{\mathbf{u}}(k) &= \mathbf{u}(k) + \mathbf{w}_{\mathbf{u}}(k) \end{aligned}$$

where M is the mass of the vehicle, c_{D} is a drag coefficient, and $\mathbf{w}_{\mathbf{u}}$ is a noise term. We assume to have $N = 25$ range-only sensors sparsely distributed at varying heights from a plane surface. The different heights simulate an uneven seafloor. A schematic representation of the simulation test case is shown in Figure 2.5.

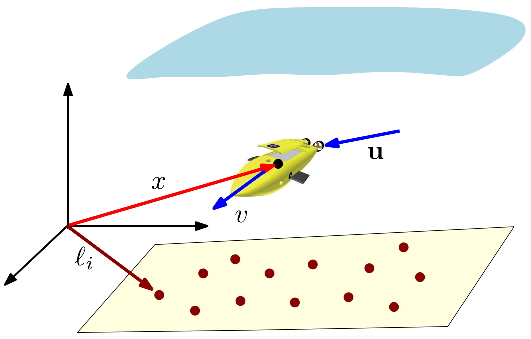


Figure 2.5: *Schematic representation of the AUV test case.*

2.4.4 Simulation Results

In this setup, we analyze and compare distributed Extended/Unscented Kalman Filters (Algorithm 2.4) and distributed Particle Filters (Algorithm 2.3), given the absence of constraints. The goal of these simulation results is to illustrate the better estimation quality of the proposed algorithms, based on the weighted consensus algorithm 2.1, with respect available methods that are based on standard consensus iterations, namely (Cattivelli and Sayed, 2010) for the UKF and (Gu et al., 2008) for the PF. Furthermore we want to show the possibilities of the underlying framework of the algorithms to handle different local estimators on different sensor nodes.

We use $\Delta t = 1$ s, $T = 130$ s, $m = 1$ kg, $C_{\mathrm{D}} = 1$ kg/s. We define an open-loop control sequence of magnitude $||\mathbf{u}(k)|| = 0.5$ N and varying direction, while we select std$(\mathbf{w}_{\mathbf{u}}(k)) = (0.05, 0.05, 0.025)^T$ N. We assume that the measurement error in equation (2.3) is std$(v_i) = 0.1$ m, for all the sensors. We consider 500 particles for the distributed Particle Filter.

In the first scenario we consider, each sensor is assumed to run the same type of local filter. We collect the results for 2500 different simulation runs, varying randomly the position and the communication range of the sensor nodes.

In order to analyze the results we utilize a metric defined on the topology of the communication graph the sensor nodes are using. In particular, this communication graph depends on the position of the nodes (and their communication range). Loosely speaking, if two sensor nodes are closer than a certain range, they can communicate, otherwise they cannot. This formally defines a graph $\mathcal{G}$ (supposed to be connected by the problem definition) and an associated Laplacian matrix L, whose second smallest eigenvalue dictates the connectivity properties of $\mathcal{G}$. If we indicate with λ_2 the second smallest eigenvalue of the graph of one simulation run, and with $\lambda_{2,\max}$ the maximum of λ_2 over all the simulation runs, then we can use $\lambda_2/\lambda_{2,\max}$ as metric to analyze our results. In fact, values of $\lambda_2/\lambda_{2,\max}$ near 1 correspond to highly connected graphs, thus estimation problems close to the centralized case, while values of $\lambda_2/\lambda_{2,\max}$ near 0 correspond to sparsely connected graphs.

Figure 2.6 depicts the average error of the the proposed algorithms versus $\lambda_2/\lambda_{2,\max}$. A dot at the coordinate (ϕ, ψ), corresponds to an average error of ψ for graphs with $\lambda_2/\lambda_{2,\max} \in (\phi - 0.05, \phi + 0.05)$. The shaded areas show the standard deviation of these errors. Note that the DEKF estimations are not depicted here because they do not converge. The DUKF are shown without the standard deviation to make the graph more readable, their values are in the order of 0.3 m.

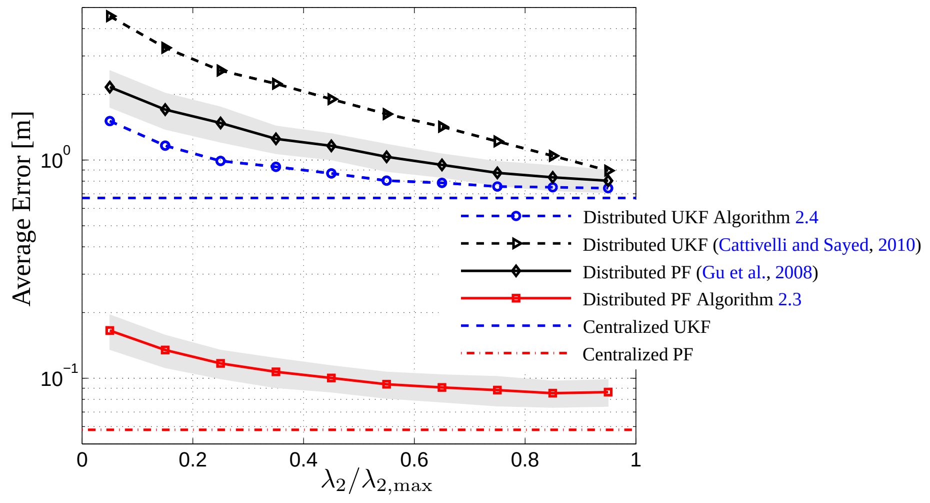


Figure 2.6: *Comparison between the proposed algorithms and the one in the literature with respect to the normalized algebraic connectivity of the information exchange graph. The average error is computed as the mean error of the sensor averaging* 2500 *different simulations. The shaded areas are the standard deviations. The DUKF's standard deviation are not depicted. The straight lines correspond to the centralized UKF, and the centralized PF.*

The results show that the proposed distributed Unscented Kalman Filter and Particle Filter outperform the ones found in the literature (Cattivelli and Sayed, 2010, Gu et al., 2008) that use a standard consensus iteration. Notice that we have chosen to compare our method only with other distributed parametric Particle Filters, since they are the only ones suitable to be employed directly in our common framework. In this context the available literature (Sheng et al., 2005, Sheng and Hu, 2005, Gu, 2007, Gu et al., 2008, Gu and Hu, 2009, Liu et al., 2009) can be divided into two parts, the first one that considers a standard consensus algorithm (2.8), whose methods can be represented with (Gu et al., 2008), and the second one, namely the work (Oreshkin and Coates, 2010) that uses a different consensus mechanism, qualitatively similar to our proposed algorithm, but optimized for an asynchronous communication case. Therefore the comparison in Figure 2.6 is representative of the current state-of-the-art of distributed (synchronous) parametric Particle Filters that have the state estimate and its covariance as input-output variables.

The reason for the difference between the methods in (Cattivelli and Sayed, 2010, Gu et al., 2008) and our proposed algorithms is due to the MERGE algorithm. In fact, this algorithm delivers estimates closer to the minimum-variance one. This also means that, for the Particle Filter, a given set of particles will characterize the a posteriori distribution better, since the trace of our covariance is smaller than in (Gu et al., 2008). A limitation of our procedure is that this ‘small covariance’ could cause an impoverishment of the particle diversity in the case of Particle Filters, which may lead to a loss in accuracy. This has not been detected in this simulation case but it may become a problem as we will see in the second test scenario below.

Our results show also that in this simulation study, the distributed Unscented Kalman Filter has a similar average error as the distributed Particle Filter reported in the literature. This is important because the distributed Unscented Kalman Filter is less computationally expensive than distributed Particle Filters, which is interesting in the context of designing fast yet accurate algorithms.

As a second test scenario, we fix the graph topology with a normalized algebraic connectivity of 0.6, and we vary the types of filters embedded on each sensor. We collect data from 1500 simulation runs, with a varying number of Particle Filters, UKFs and EKFs present in the sensor network and their physical location. Figure 2.7 summarizes our results. The curves represent different numbers of Particle Filters. Since the overall number of sensors is fixed ($N = 25$), one can compute the number of EKFs present in the network from the knowledge of the number of Particle Filters and UKFs.

We can observe that even with a relatively small number of more accurate filters (for example 1 PF and 5 UKF), the distributed estimation converges. This was not the case in the first test scenario, where the local estimates were diverging using the EKFs alone. This is a very interesting observation that seems to support having many cheap devices and only a very few expensive ones.

We may also notice that by increasing the number of UKFs, the accuracy improves initially quite noticeably but eventually deteriorates. For a low number of UKFs, this trend can be explained by the merging mechanism as well, as illustrated in Remark 2.4.

Remark 2.4 *Let n_{UKF} be the number of UKFs and n_{PF} the number of the Particle Filters, and let the covariance of the filters be $\bar{P}_{\text{PF}}$, $\bar{P}_{\text{UKF}}$, and $\bar{P}_{\text{EKF}}$ respectively. Assume for simplicity, a scalar state vector. The average error is then related to the merged covariance, whose expression is*

$$\hat{P} = \frac{\bar{P}_{\text{PF}}\bar{P}_{\text{UKF}}\bar{P}_{\text{EKF}}}{25\bar{P}_{\text{PF}}\bar{P}_{\text{UKF}} + \bar{P}_{\text{UKF}}(\bar{P}_{\text{EKF}} - \bar{P}_{\text{PF}})n_{\text{PF}} + \bar{P}_{\text{PF}}(\bar{P}_{\text{EKF}} - \bar{P}_{\text{UKF}})n_{\text{UKF}}},$$

which for a given n_{PF} is proportional to

$$\hat{P} \propto \frac{1}{\alpha + n_{\text{UKF}}}$$

with the constant $\alpha > 0$. This explains the initial decrease of the average error for an increasing number of UKFs.

The deteriorating performance for a higher number of UKFs is instead an open question. One potential reason is the suboptimality of the merging mechanism and the ‘small covariance’ problem. In fact, the proposed algorithm may lead to a smaller covariance estimate than the actual one, which leads to optimistic filters (Bar-Shalom et al., 2001), and this could cause a poor selection of the sigma points for the UKFs. Adding more EKFs increases the average error and brings the estimated covariance closer to the real one, which improves performance.

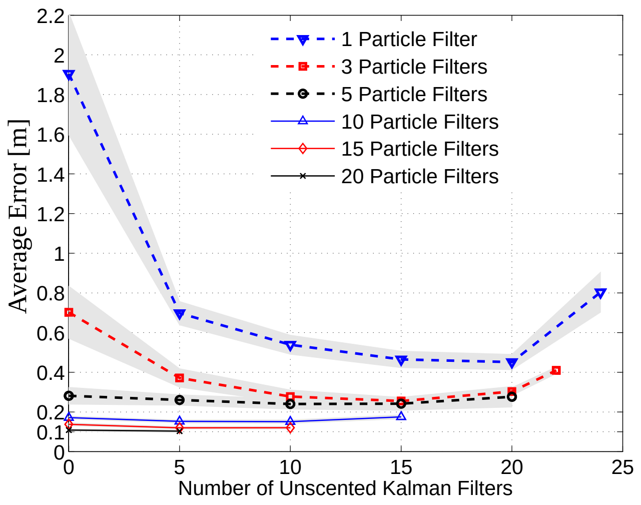


Figure 2.7: *Results for different filters running on different sensors. The Average Error is computed as the mean error of the sensor averaging* 1500 *different simulations. The shaded areas are the standard deviations of the bold lines. Since the number of sensor is fixed (*$N = 25$*), one can compute the number of EKFs present in the network from the knowledge of the number of Particle Filters and UKFs.*

2.5 Conclusions

In this chapter we have studied the distributed nonlinear state estimation problem and we have proposed distributed nonlinear estimators that leverage on common underlying framework. This framework is based on a weighted consensus mechanism and could allow the usage of different estimators on different sensor nodes, which is an important aspect when considering heterogeneous sensor networks. Simulation results have illustrated the benefit of this framework with respect to standard distributed algorithms and how its performance relates to centralized estimators.

2.6 Open Problems and Future Work

Distributed Estimation is an area that is still evolving and many challenges are still open. We can highlight two interesting problems that require further attention as follows.

Constrained Consensus

The first problem is the design of consensus algorithms that can handle generic non-convex constraints.

As remarked, consensus algorithms make use of convex combinations of the initial values to reach an agreement among the different nodes of the sensor network. For this reason, when there are non-convex constraints on the variable to agree upon, typical consensus algorithms may deliver an unfeasible final value. This is a fundamental limitation of current consensus schemes that affects all the distributed estimation algorithms. This is also the reason why there are still no distributed methods to tackle the distributed estimation problem of Section 2.2.1 in its full generality. We refer the reader to the interesting work in (Nedić et al., 2010) where constrained consensus algorithms are introduced in a convex setting. We believe that, if the methods presented in (Nedić et al., 2010) could be developed further to address non-convex constraints, they might be extremely helpful in devising new and more general distributed methods for the distributed estimation problem.

Reachability Problem in the distributed MHE

The second problem we remark is the reachability problem in the distributed MHE formulation.

As illustrated in the chapter, due to the nonlinear nature of the system dynamics, the agreement on the arrival cost could deliver an unreachable state and therefore it could drive the local estimator towards inadmissible regions of the state space. One solution to this problem could be to eliminate the nonlinear dynamic model using differential flatness techniques as in (Mahadevan and Doyle III, 2004) for differential flat systems. For general nonlinear systems the problem is instead more difficult to handle.

Chapter 3

Distributed Computation Particle Filters on GPU-Architectures

Abstract — In this chapter we consider methods to reduce the possibly high computational requirements of nonlinear estimators by distributing the computations among different computing devices communicating one another.

In particular, we study how to implement Particle Filters using GPU-architectures, for real-time control or monitoring applications. Experimental results on a robotic arm will illustrate that the concept of fast yet accurate nonlinear filtering is possible by a suitable adaptation of the Particle Filter algorithm.

3.1 Introduction

In Chapter 2 we have seen approaches to distributed estimation of a given nonlinear process in sensor networks, where sensor nodes have access only to local measurements but they can communicate with the neighboring devices. It is also well-known that accurate nonlinear estimation can be very demanding in terms of high computational requirements. We use as an example the Particle Filters, which in the nonlinear/non-Gaussian setting usually outperform Kalman Filter type methods, but suffer from very high computational requirements when using a high number of particles. This aspect is particularly important in sensor networks, where the individual sensor nodes have typically a limited amount of computational resources, but it is also very important for centralized settings.

The topic of this chapter is to propose a distributed computation approach for Particle Filters, in order to achieve real-time and accurate estimation. In particular, we will make use of GPU[1]-architectures to distribute the computations of Particle Filters among different computational units. This idea is motivated by the recent the rise of reasonably priced graphical processing units featuring thousands of GPU cores that compete with and take the

[1]GPU stands for Graphical Processing Unit usually composed of many computing cores that work in parallel on data-parallel tasks such as graphics rendering. We remark that for the main message of the chapter, it is not important to know how a GPU works in practice. As a matter of fact, our scheme could be implemented on any systems with multiple cores that communicate with each other.

place of the best traditional desktop single-core CPU processing architectures in specific types of computations. In this context, one could imagine thousands of particles running on each one of the thousand cores.

The main question we will pose and answer is the following: *can Particle Filters be run efficiently enough and deliver accurate estimates to be implemented in high sample rate real-time feedback control applications?*

In particular, after surveying the available techniques to distribute the computations among the computation units, in Section 3.2 we will propose an approach that considers a Particle Filter as a network of smaller filters, where each of them exchanges data locally based on the network topology. Finally, we will test our implementation in numerical simulations and on a robotic arm experimental setup, where we demonstrate 100 Hz real-time feedback control based on a Particle Filter using more than a million particles.

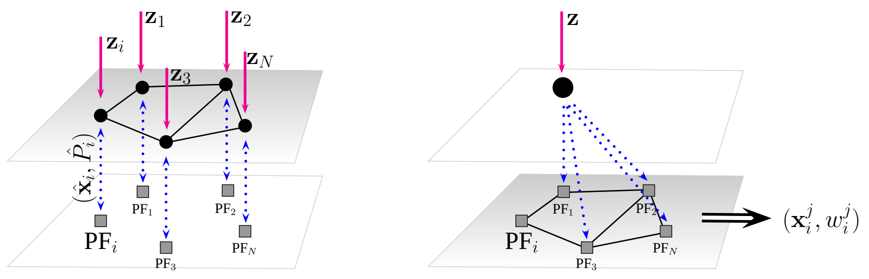
We remark that the main contribution of this chapter will be the proposed distribution of the computations among the different computing units. This proposed idea will be shown to outperform standard implementations based on parallel computing (instead of distributed ones). In particular, we will be able to increase the number of particles, the sampling frequency, and the state dimension often by orders of magnitude with respect to state-of-the-art GPU solutions. Furthermore, we will show that our scheme has comparable accuracy with centralized sequential particle filters (with the same number of total particles), which require 10-100 times more computational time (when using a high number of particles) than our proposed distributed implementation.

3.2 Distributed Computation Framework

In this chapter we will use the same notation as Chapter 2, Section 2.3.2 (where we have discussed centralized Particle Filters), but we will consider their distributed *Computation* implementation. We refer the reader who is not familiar with Particle Filters to their detailed introduction in Section 2.3.2.

Formally, we define Distributed Computation Particle Filters as the ones that have access to all sensor measurements but use only a subset of particles in each computing unit. The different units where the distributed Particle Filters are running are depicted in Figure 3.1.b (we have reported in Figure 3.1.a the distributed (sensing) setting for reference). Since the different local particle filters (PF) have the same measurement vector, they can exchange directly particle-weight couples, $(\mathbf{x}^j, w^j)$ (since the weights have been weighted on the same measurement vector). The distributed estimation is done in this case at the level of the filters whereas the sensors are only a means through which the measurements become available. This is the class of estimators that we will study in this chapter.

In the following, we will review the body of algorithms that can be considered a Distributed Computation Particle Filter. It is somewhat surprising that the exploitation (and design) of the communication network among the units, as sketched in Figure 3.1.b, is a concept that is rather absent in the reviewed literature (where instead a traditional parallel strategy is often used). This concept, extensively used in the sensor network community, is one of the main ingredients that will enable us to devise more efficient Particle Filters.



(a) Distributed (sensing) approach (b) Distributed Computation approach

Figure 3.1: *Distributed Computation Particle Filters and their relationship with Distributed (sensing) Particle Filters of Chapter 2, Section 2.3.2. In the Distributed (sensing) setting, the exchange of information is done sharing the local means and covariances of the state estimates* $(\hat{\mathbf{x}}_i, \hat{P}_i)$. *In the Distributed Computation setting, the distribution and communication is done via particle-weight couples. The final outcome is a number of different a posteriori distributions represented via* $(\mathbf{x}_i^j, w_i^j)$. *The fact that these distributions are different means that the different local particle filters do not necessarily have to agree on a common one.*

3.2.1 Related Work

In the past decade, with the rise of the massive parallelization made possible by GPUs, many researchers have analyzed, studied, and designed versions of distributed Computation Particle Filters. These algorithms differ in the number of particles they can handle, the specific parallelization, and the degree of communication between the computing units. In the next paragraphs we will examine a number of strategies to implement distributed and parallel Particle Filters.

To the author's best knowledge, the first work dealing with parallelism in Particle Filters is (Brun et al., 2002). In this paper, the particle population is partitioned into several subsets, each assigned to a separate processor. Sampling, weight calculations, and resampling are performed independently and locally for each subset. The authors consider the weighted sum of all the particles as the estimate. This estimation is achieved by calculating for each subset a local estimate and a local sum of weights which are, subsequently, gathered centrally and combined into a global estimate. The authors show that local resampling is comparable with global resampling, in terms of estimation error.

In the work (Bashi et al., 2003), three methods are proposed to implement distributed computation Particle Filters: *(i)* Global Distributed Particle Filter (GDPF), *(ii)* Local Distributed Particle Filter (LDPF), and *(iii)* Compressed Distributed Particle Filter (CDPF). With GDPF, only the sampling and weight calculation steps run in parallel on different processors, while resampling is performed centrally. All particle data is transferred to a central unit for the resampling step and the new particles are sent back to each processor. The central unit calculates the global estimate from the particle data. With LDPF, resampling is also performed locally on each processor without any communication with other processors. Aggregated particle data is sent to a central unit in order to calculate the global estimate similar to the algorithm of (Brun et al., 2002). In CDPF, similar to GDPF, resampling and the calculation of the global estimate are performed centrally, but only a small representative subset of the particles of each processor are sent to the central unit. The paper concludes from a number of simulations that LDPF provides both better estimation and performance.

Two distributed computation Particle Filter algorithms are proposed in (Bolić et al., 2005): *(i)* Resampling with Proportional Allocation (RPA), and *(ii)* Resampling with Non proportional Allocation (RNA). Both algorithms perform the sampling and weight calculation stages in parallel. In the RPA method the resampling stage involves centralized communication, whereas in the RNA method it is performed completely locally. Different particle exchange mechanisms are discussed to improve the performance of this local resampling step, but it is rather unclear how these particles are selected. Furthermore, the amount of particles that are exchanged among the cores is a significant ratio of the total population (at least 25% of all particles of each processing element). In both cases (RPA and RNA) the estimate is calculated as the weighted average of all particles from all processing elements. It is argued that RPA provides a better estimation, while RNA has a simpler design. In a later work (Bolić et al., 2010), the authors compare a standard Particle Filter with a Gaussian Particle Filter on an FPGA. The presented results indicate that the Gaussian Particle Filter, while being faster than a standard Particle Filter, is equally accurate for (near-)Gaussian problems.

A number of the previously presented algorithms (GDPF, RNA, RPA, Gaussian particle filter) are compared using a parallel implementation on a multi-core CPU for a bearings only tracking experiment in (Rosén et al., 2010). The comparison goes only until 10K particles. As expected, the Gaussian particle filter outperforms (in terms of accuracy over computational time) all other algorithms, since the estimation problem is Gaussian. The other Particle Filter algorithms (GDPF, RNA, RPA) exhibit similar estimation accuracy. In terms of runtime performance, both RNA and the Gaussian Particle Filter achieve near linear speedup with respect to the number of cores for a large number of particles, while GDPF and RPA exhibit only sub-linear speedup.

An interesting Particle Filter implementation is presented in (Hendeby et al., 2010), where the authors exploit GPU specific hardware features. In this paper, first, a parallel approach for sampling and weight calculations is proposed and then, the resampling step is performed using a specific hardware feature of GPU's called the rasterizer. In practice this step is close to the RNA algorithm of (Bolić et al., 2005) but, since pseudo-random numbers are generated on the host CPU and successively transferred to the GPU, the performance of the filter is severely damaged. In fact, about 85% of the total runtime is spent on generating pseudo-random numbers and transferring them to the GPU, making this implementation not suitable for real-time estimation in complex problems.

The GPU implementation described in (Chao et al., 2010) consists of parallel sampling, parallel weight calculations, and resampling performed locally on the different computing units. For the sampling step, the authors propose to use the finite-redraw importance-maximizing (FRIM) method, which checks the weight of the drawn particle and redraws until a particle with a reasonable weight is constructed. We note that the FRIM method is known to reduce the required total number of particles, but a fixed number for maximum number of redraws has to be imposed to limit the iterations. The generation of random numbers is performed on the host CPU, as in (Hendeby et al., 2010), and subsequently copied into the GPU. This makes their presented implementation rather limited. In fact,

Table 3.1: *Available methods to implement distributed computation Particle Filters.*

Ref.s		Sampling + weight	Resampling	Estimation	Particles	State dimension	Runtime [ms]
Brun et al. (2002)		local	local	central	32K	3	1 – 100
Bashi et al. (2003)	GDPF	local	central	central	5K	5	100
	LDPF	local	local	central	5K	5	10
	CDPF	local	central§	central§	5K	5	10
Bolić et al. (2005)	RPA	local	central§	central	50K	3	1
	RNA	local	local	central	50K	3	0.1 – 1*
Hendeby et al. (2010)		local	central§	central	1M	2	1000
Chao et al. (2010)		local	local	unknown	4K	3	200
Par and Tosun (2011)		local	central	central	130K	4	10

§ only part of the particles are sent to (multiple) computing hubs.

* these are theoretical limits based on the considered hardware rather than a measured performance.

with the use of a low performance laptop-GPU, they are able to run experiments only up to 4K particles with execution times around 200 ms in the best case. It is unclear how the estimate is calculated from the weighted particle set and whether it is executed on the GPU.

The recent study (Par and Tosun, 2011) investigates a Particle Filter for localization and map matching for vehicle applications on a CPU using OpenMP and on a GPU using CUDA. The state dimension is only four and the estimation does not benefit from more than 32K particles, but the application is nevertheless an interesting and well-explained case for Particle Filters. Experiments show that, with 128K particles, a CPU is 4.7 times faster on six cores than a single CPU architecture, while a GPU is another 16 times faster. The proposed algorithm runs parallel sampling and weight calculations but the resampling step is done on the host CPU in a centralized fashion. However, the resampling is performed only when the particle variance is above a threshold. This scheme seems to be faster on average than the other mentioned algorithms but it suffers from high peaks of computation time when the resampling is performed, which is undesirable for real-time applications.

Table 3.1 summarizes the surveyed methods and highlights the degree of centralization still presented in many of them. In particular, the resampling stage is performed either in a centralized fashion (where all or a part of the particles of each core are sent to some computing hub), or locally, without exchange of particles. This way of operating is typical from a parallel perspective.

We note however that these ways to resample the particle population can degrade the performance of the filter (in terms of computational time and accuracy) rather significantly. In order to address this problem we will introduce in the next section the concept of distributed resampling, which will enable us to overcome these difficulties and achieve an improvement over the aforementioned methods. Furthermore, we will explain how the different (user-tunable) parameters can affect performance.

3.2.2 Proposed Approach

In this section we present our proposed approach to implement distributed computation Particle Filters. First we give a brief introduction to GPU architectures and we introduce the concept of topology. Then, we describe the distributed resampling techniques that is the core of our method, and, finally, after presenting our algorithm, we analyze the selection of its (user-tunable) parameters.

GPU Architecture and Topology

GPU's are programmed to exploit their inherent parallelism to execute at the same time a significant number of tasks. We utilize CUDA (NVIDIA, 2010) as a programming interface. A CUDA application is divided into a *host* and *device* side. The host refers to the CPU which is connected to one or more devices (i.e., the GPUs). The host manages device memory, initiates data transfers and launches kernels on the device. Kernels are special functions executed on the device in parallel. Each kernel typically consists of numerous threads grouped into thread blocks. Limited fast access shared memory is available to all threads from a single block for local communication while slower access global device memory is available to all threads. The thread groups and the host can access the global device memory and this is typically the way the data is shared (although it is also the main cause of bottlenecks in standard implementations).

Figure 3.2 depicts the terminology and the architecture. For our implementation, each thread is particle, while each thread block is a local Particle Filter.

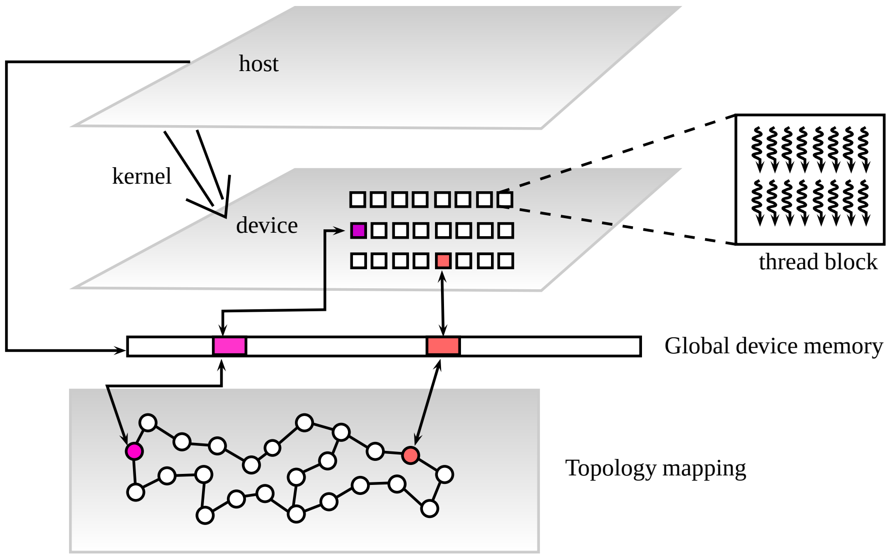
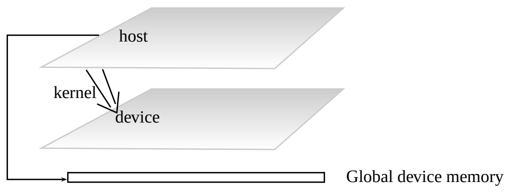


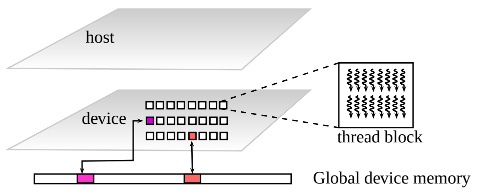
Figure 3.2: *Basic concept of* GPU *architectures as available in CUDA. The global device memory is mapped into a specific topology representing data-exchange between thread blocks.*

In particular, with reference to Figure 3.2,

The host launches kernels. Each kernel is a specific function, such as “sampling”, “sorting”, or “resampling”. The host has access to the global device memory.



Kernels consist of numerous threads grouped into thread blocks. For us the thread blocks represent the local particle filters, while the threads the particles. Different threads have access to fast limited shared memory inside the thread block and they can write and read data from the slower access global device memory.



In Figure 3.2, we also illustrate the concept of topology, which will be important for our implementation. Since the access to the global device memory is usually a bottleneck, it is important to limit this operation and *group* the data that each thread block needs to read. Our idea is to map the global device memory into a specified topology, that formally defines this grouping procedure. Using Graph Theory terminology, each thread block is a node, while if two nodes can access each other’s data we say that there is an edge between them. The set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}$ define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with specified topology. Since given the graph $\mathcal{G}$ the topology is fixed, we often use the symbol $\mathcal{G}$ to refer to both interchangeably. In this context, each thread block has a set of neighbors, i.e., the thread blocks it can share the data with.

We consider each kernel to consist of N thread blocks (later local filters) labeled with the index $i = 1, \dots, N$, whereas each thread block has m threads (later particles), labeled with the index $j = 1, \dots, m$. The number of neighbors of each thread block is indicated with N_i.

Moreover, as a further abstraction of the hardware/software level, we refer to the thread blocks as computing units that are able to send and receive data from the neighboring computing units, via the graph $\mathcal{G}$. Since the access to global device memory is often a bottleneck and since the local shared memory is limited by the hardware, the communication among the computing units cannot grow arbitrarily.

Distributed Resampling

In typical implementations of distributed computation Particle Filters, the sampling and weight computation steps are done locally (in parallel) in a rather straightforward fashion. We also follow this standard strategy. The resampling step is instead more delicate.

Resampling is a critical step in Particle Filters. On one side, it is necessary since it can provide the chance for good particles to spread themselves and it overcomes the degeneracy phenomenon, where after a few iterations, all but one particle will have negligible weights. However, on the other side it introduces practical (and often application-specific) issues, notably the impossibility to run the code in parallel, since all the particles must be combined, and the loss of diversity among the particles as the resultant samples contain many repeated points. In this context, it is rather crucial to devise carefully the resampling stage and give to the user tunable parameters to overcome the mentioned application-specific issues.

As surveyed in Section 3.2.1 the resampling step is often performed with a degree of centralization. Typically all the particles or (a significant) part of them are sent to a limited number (often one) computing hubs (i.e., thread groups).

In contrast with these methods, we propose a distributed approach to the resampling stage. Our idea is rather simple yet extremely effective and it is based on the considerations that there is no need to have specified computing hubs, each of the computing units can serve as a computing hub itself, and there is no need to send all the particles, in fact just very few are necessary to the resampling step. In our approach each computing unit sends to *its neighbors* only t representative particles (the ones with locally highest weights) and performs the resampling stage on its resulting $m + tN_i$ particles (we recall that N_i is the number of neighbors a computing unit has). We note that this simple idea is extremely powerful. In fact:

1. The number of shared particles is a small part of the population, since typically $tN_i \ll m$. However, as we will see in the experimental and simulation tests of Section 4.2.7, the fact that the method is not completely local (meaning $t > 0$, in contrast with local methods where $t = 0$, for example (Brun et al., 2002, Bolić et al., 2005, Chao et al., 2010)) increases significantly the accuracy of the filter.

2. There is no need of centralized data collection, making the resampling step fast and efficient.

3. Both the topology and the number of shared particles t are user parameters that can be adjusted to the application at hand. In some cases (for example, in high process noise setting), we will see that having an all-to-all topology (similar to the centralized hub in the CDPF algorithm of (Bashi et al., 2003)) could lead to worse performance than a ring topology. As we will explain below and in Section 4.2.7 this is due to the loss of diversity introduced by the resampling stage.

This idea of distributed resampling has been presented by in (Simonetto and Keviczky, 2009). A similar approach has been also presented in the later work (Balasingam et al., 2011), whose authors propose a modification of the distributed resampling idea of (Simonetto and Keviczky, 2009) on a ring topology, where the local computing units substitute their highest weight particles with the ones of the neighboring units. Both these works show in simple numerical simulations the similar performance of the distributed strategy with a standard single-core CPU implementation that uses the same number of particles Nm, in terms of estimation quality.

In this chapter, we implement the distributed resampling scheme on a GPU architecture and we test it on a real hardware platform. Furthermore, we analyze the effect of the parameters of algorithm, namely the number of shared particles t and the topology $\mathcal{G}$ on the quality of the estimate.

Proposed Algorithm

Our proposed algorithm consists of the following high-level steps:

a. Sampling and weight calculation: this step is done locally on each computing unit;

b. Distributed resampling step: this step is done in a distributed computation fashion as previously explained;

c. Local estimation: this step is done locally on each computing unit, picking the local particle with maximum weight.

Algorithm 3.1 describes more in detail how these three high-level phases are translated into high-level commands. For the specific hardware implementation, the reader is referred to (Chitchian et al., 2012b).

Remark 3.1 *In addition to Local estimation, we also provide to the user a Global estimate kernel which selects the best particle among the local bests. This selection is also done in parallel via a parallel reduction on the winners from each block. In our experiments we have noted that the extra runtime spent in the Global estimation kernel is extremely limited compared to the other kernels.*

Algorithm 3.1 Distributed Computation Particle Filter:
high-level description on each computing unit i

```
Input: $\{\mathbf{x}_i(k-1)^j\}_{j=1,\dots,m}, \mathbf{z}(k)$
▷ Available Data: $p(\mathbf{x}(k)|\mathbf{x}(k-1)), p(\mathbf{z}(k)|\mathbf{x}(k)), t, m$, topology $\mathcal{G}$
1. Local Filter:
    for j = 1:$m$
        1.1: sampling: $\mathbf{x}_i(k)^j \sim p(\mathbf{x}_i(k)|\mathbf{x}_i(k-1)^j)$
        1.2: weight_calculation: $w_i(k)^j = p(\mathbf{z}(k)|\mathbf{x}_i(k)^j)$
    end
2. Sorting: sort $\{\mathbf{x}_i(k)^j\}_{j=1,\dots,m}$ according to $\{w_i(k)^j\}_{j=1,\dots,m}$
3. Estimation: local_estimation: pick $\mathbf{x}_i(k)^j$ with maximal $w_i(k)^j$
4. Particle Exchange:
    foreach neighbor do
        send and receive $t$ particle-weight couples to/from neighbors
    end
5. Resampling: resample the $m + N_i t$ particles into $m$ particles
6. Reset: set $w_i(k)^j = 1/m$ for all $j$
Output: $\{\mathbf{x}_i(k)^j\}_{j=1,\dots,m}$
```

Analysis of the Algorithm

In this section we provide some insights in the selection of the parameters t and $\mathcal{G}$ of Algorithm 3.1, while, in general, the selection of N and m are dictated by hardware limitations.

Let $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ be the local approximation of the a posteriori PDF, different for each local filter, which can be written as

$$\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k)) = \frac{1}{\Omega_i(k)} \sum_{j=1}^{m} w_i(k)^j \delta(\mathbf{x}(k) - \mathbf{x}_i(k)^j), \tag{3.1}$$

where the subscript i indicates that weights and particles are referred to the local filter i, and $\Omega_i(k) = \sum_{j=1}^{m} w_i(k)^j$. Let $\tilde{w}_i(k)^j$ be the weight of particle j of filter i after the communication step but before resampling. Let $\tilde{m}_i = m + N_i t$ be the total number of particles of filter i before resampling, and let $\tilde{\Omega}_i(k) = \sum_j \tilde{w}_i(k)^j$.

First of all, we support the intuitive claim that the most representative particles of the $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ in (3.1) are the ones with highest weights, which justifies the communication strategy in Algorithm 3.1. In order to show this, we utilize the Kullback-Leibler (KL) divergence (Cover and Thomas, 1991) that measures the distance between two PDFs. In particular, the smaller the KL divergence is, the closer the two PDFs are. Consider the approximated a posteriori $\hat{p}_i^{(t)}(\mathbf{x}(k)|\mathbf{z}(k))$ computed using only $t < m$ particles, as

$$\hat{p}_i^{(t)}(\mathbf{x}(k)|\mathbf{z}(k)) = \frac{1}{\Omega_i^{(t)}(k)} \sum_{j=1}^{t} w_i(k)^j \delta(\mathbf{x}(k) - \mathbf{x}_i(k)^j) \tag{3.2}$$

with $\Omega_i^{(t)}(k) = \sum_{j=1}^{t} w_i(k)^j$, then the claim that the t particles with highest weights are the most representative for (3.1) is formally expressed as follows.

Proposition 3.1 (Balasingam et al., 2011) *The KL divergence between* (3.1) *and its approximation* (3.2)*, which employes only $t < m$ particles, i.e., $D(\hat{p}_i, \hat{p}_i^{(t)})$, can be written as*

$$D(\hat{p}_i, \hat{p}_i^{(t)}) = -\log\left(\sum_{j=1}^{t} \frac{w_i(k)^j}{\Omega_i(k)}\right) = -\log\left(\frac{\Omega_i^{(t)}(k)}{\Omega_i(k)}\right) \tag{3.3}$$

and it is minimal when we use for (3.2) *the t particles with highest weights.*

We can distinguish two main aspects that affect performance of Algorithm 3.1. First, we have to analyze how good each of the local a posteriori $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ represents the global $\hat{p}(\mathbf{x}(k)|\mathbf{z}(k))$. This aspect dictates the estimation quality of the distributed computation particle filter with respect to a standard single-core CPU implementation. Second, it is important to study how distorted each of the local $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ becomes after the resampling step. This distortion is a measure of the distance between the resampled and the initial population. In particular, high values of distortion generally mean that the filter will be affected by the degeneracy/loss of diversity phenomenon, where only few particles have non-zero weight.

In order to analyze the quality of each local a posteriori $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ with respect to the global $\hat{p}(\mathbf{x}(k)|\mathbf{z}(k))$ we make use of Proposition 3.1 applied to these distributions and we define the cumulative sum of the local KL divergences as

$$\begin{aligned}\sum_{i=1}^{N} D(\hat{p}, \hat{p}_i) &= -\sum_{i=1}^{N} \log \left(\sum_{j=1}^{\tilde{m}_i} \frac{w_i(k)^j}{\Omega(k)} \right) \\ &= -\sum_{i=1}^{N} \log \left(\frac{\Omega_i(k)}{\Omega(k)} + \sum_{j=m+1}^{m+N_i t} \frac{w_i(k)^j}{\Omega(k)} \right), \end{aligned} \tag{3.4}$$

where we recall that $\Omega(k) = \sum_{j=1}^{Nm} w(k)^j$.

Let $W = \sum_{j=m+1}^{m+N_i t} w_i(k)^j$. By the fact that we are sharing the t particles with highest weight, we could approximately consider $w_i(k)^j$ to be the same for each j, i.e., $w_i(k)^j \approx w_i(k)^{(t)}$, and approximate W as

$$W \approx w_i(k)^{(t)} N_i t.$$

Consider the derivative of the cumulative sum (3.4) respect to W (or $N_i t$) as

$$\frac{\partial}{\partial W} \sum_{i=1}^{N} D(\hat{p}, \hat{p}_i) = -\frac{\Omega(k)}{\Omega_i(k) + W}, \tag{3.5}$$

which is minimal for $W = 0$.

From the relations (3.4) and (3.5) we can infer the following.

- The higher $N_i t$ is, the closer the local and global a posteriori are. This follows from (3.4) with $W \to \infty$ (or in the approximated sense, with $N_i t \to \infty$).

  Therefore, increasing the communication leads to an increase of estimation quality for a given time step k (note that the effect on the time step $k+1$ depends also on the resampling, which is analyzed next). In particular, if we choose all-to-all communication and $t = m$, then each local population is comprised at least of the m particles with highest weight, which are the most representative to describe $\hat{p}(\mathbf{x}(k)|\mathbf{z}(k))$.

- The gain in increasing $N_i t$, or for a given N_i, in increasing the number of share particles t, is maximum when $W = 0 \approx N_i t$. In other words, we can expect a more significant increase in the estimation quality passing from $t = 0$ to $t = 1$ than passing from $t = 1$ to $t = 2$.

Besides choosing t and N_i (and therefore the topology $\mathcal{G}$) to minimize the KL divergence between the local $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ and the global $\hat{p}(\mathbf{x}(k)|\mathbf{z}(k))$ while maintaining $N_i t$ as small as possible to limit the communication effort, the effect of the resampling stage is also an important aspect to consider. Assume that $N_i t \ll m$ and thus consider the local particle population to be uncorrelated among the local filters i. Define the distortion (Míguez, 2007) of $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ after the resampling step as its KL divergence with the global a posteriori $\hat{p}(\mathbf{x}(k)|\mathbf{z}(k))$. We note that this measure is different from the one in Equation (3.4), since we use for $\hat{p}_i(\mathbf{x}(k)|\mathbf{z}(k))$ the resampled weights (before resetting them). The dependence of the distortion on the local weights before resampling $\tilde{w}_i(k)^j$ can be approximated as (Míguez, 2007)

$$\mathcal{D}_{\tilde{w}} \approx \sum_{i=1}^{N} \Omega_i(k) \sum_{j=1}^{\tilde{m}_i} \left\lceil \frac{\tilde{w}_i(k)^j}{\Omega_i(k)} \right\rceil \log \left(\left\lceil \frac{\tilde{w}_i(k)^j}{\Omega_i(k)} \right\rceil \right). \tag{3.6}$$

In order to avoid distortion and therefore in order to maximize the estimation quality, $\mathcal{D}_{\tilde{w}}$ has to be minimized, which is achieved when the different $\tilde{w}_i(k)^j$ within the same local filter i have similar values. On the contrary when few particles have the highest weights, the distortion is close to its maximum and therefore degeneracy can be expected to occur. In particular we can distinguish two extreme cases:

- Relatively low process noise but high measurement noise. In this case the particles have similar weights and therefore the resampling step does not cause a high level of distortion, even for $N_i t \neq 0$.
- Relatively high process noise but low measurement noise. In this case few particles have the highest weight and therefore the resampling step causes high level of distortion. In particular, given t, the higher the N_i is, the higher the distortion is.

In the next section we will illustrate in practice the insights on the selection of the parameters t and N_i, which we have presented in this section.

3.3 Numerical and Experimental Results

3.3.1 The Robotic Arm Model

In order to test, verify, and benchmark our distributed computation particle filter implementation we use the realistic industrial application of a robotic arm. The main reason for such a choice is that the measurement equations of this application are highly nonlinear and extremely challenging for standard estimation techniques both for accuracy and computational time.

The robotic arm, in this experiment, has a number of joints $J = 3$ which can be controlled independently. It has one degree of freedom per joint plus the rotation of the base. Each joint has a sensor to measure its angle. There is a camera mounted at the end of the arm. This camera is used for tracking an object which is moving on a monitor on a fixed $y - z$ plane. The real robotic arm as well as a schematic representation are shown in Figure 3.3.

Let $\theta_i(k)$ be the angle of the joint i at the discrete time k ($i = 0$ represents the rotational degree of freedom of the base). Let $\mathbf{p}_{\mathrm{w}}(k) = (x(k), y(k), z(k))^\top \in \mathbb{R}^3$ be the position of the object to be tracked at the discrete time step k in the fixed reference system of the robotic arm, as indicated in Figure 3.3, while let $(v_x(k), v_y(k), v_z(k))^\top \in \mathbb{R}^3$ be its velocity. We consider $x(k)$ to be known a priori and $v_x(k)$ to be zero for all k. Denote with

$$\mathbf{x}(k) = (\theta_0(k), \dots, \theta_J(k), y(k), z(k), v_y(k), v_z(k))^\top$$

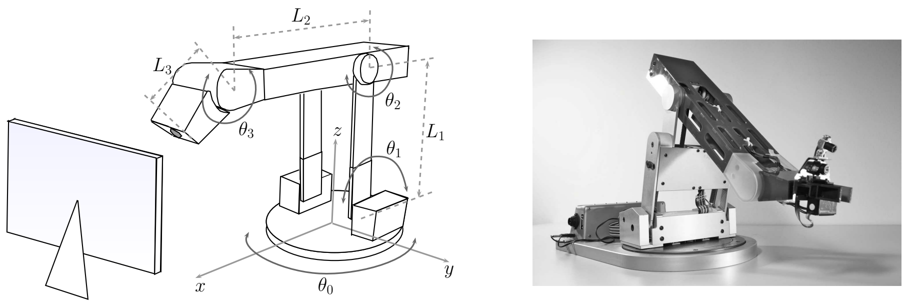


Figure 3.3: *Robotic arm used for the experimental test. On the right the real test-bed and on the left a schematic representation showing the camera at the end effector, marked with a grey circle, and the monitor on which a moving object is displayed.*

the state of the arm and object dynamics. We model the angle dynamics as discrete-time single integrators, while the object dynamics as a discrete-time double integrator:

$$\theta_i(k) = \theta_i(k-1) + \mathbf{w}_{\theta_i}(k-1), \quad i = 0, \dots, J \tag{3.7a}$$
$$y(k) = y(k-1) + v_y(k-1)\Delta t + \mathbf{w}_y(k-1) \tag{3.7b}$$
$$z(k) = z(k-1) + v_z(k-1)\Delta t + \mathbf{w}_z(k-1) \tag{3.7c}$$
$$v_y(k) = v_y(k-1) + \mathbf{w}_{v_y}(k-1) \tag{3.7d}$$
$$v_z(k) = v_z(k-1) + \mathbf{w}_{v_z}(k-1) \tag{3.7e}$$

where the terms $\mathbf{w}$ model the process noise and Δt is the sampling time. The system of dynamical equations (3.7) will represent our a priori distribution $p(\mathbf{x}(k)|\mathbf{x}(k-1))$, as explained in Section 2.3.2, Equation (2.26).

The camera mounted at the end effector of the robotic arm detects the object displayed on the monitor in its own frame of reference. Let $\mathbf{p}_\text{s}(k) = (x_\text{c}(k), y_\text{c}(k))^\top$ be the position of the object in the camera moving frame at the discrete time k. This position is measured in pixel. To relate $\mathbf{p}_\text{s}(k)$ to the actual coordinates of the object in the robot fixed frame we have first to use a camera model that translates the pixels into meters and then performs a chain of translations and rotations to change the reference frame. The camera is modeled by the traditional pinhole projection with added radial lens distortion, see (Bouguet, 2010, Hutchinson et al., 1996, van der Lijn et al., 2010) for details. The model for the measured observations of the moving object is the composition of three classes of maps: rigid body transformations, projections, and "distortion" maps. We emphasize the first two, since the lens distortion is known, i.e. the camera is calibrated a priori. Let $\mathbf{p}' = (x', y', z')^\top$ be a three-dimensional point described in generic coordinates. Let $\varphi : \mathsf{SE}(3) \times \mathbb{R}^3 \to \mathbb{R}^2$ be the standard rigid body transformation

$$\varphi(R, \mathbf{p}', \mathbf{q}) = R\mathbf{p}' + \mathbf{q},$$

where R and $\mathbf{q}$ are the traditional rotation matrix and translation vector, respectively. The camera pinhole projection model is realized by the projection map $\pi : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\pi(\mathbf{p}') = \frac{1}{z'} \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

The composition of the maps is described graphically by the informal diagram:

$$\begin{tikzcd} \underset{\text{world}}{\mathbf{p}_\text{w}} \arrow[r, leftrightarrow, "\varphi"] & \underset{\text{camera}}{\mathbf{p}_\text{c}} \arrow[r, "\pi"] & \underset{\text{plane}}{\mathbf{p}_\text{p}} \arrow[r, leftrightarrow, "\psi"] & \underset{\text{sensor}}{\mathbf{p}_\text{s}} \end{tikzcd}$$

where ψ describes lens distortions. The full sensor model is described by

$$\mathbf{p}_\text{s} = \psi \circ \pi \circ \varphi(R, \mathbf{p}_\text{w}, \mathbf{q}) + \mu_\text{s}, \tag{3.8}$$

with μ_s an additive noise term. We note that the couple $(R, \mathbf{q})$ depends nonlinearly on the configuration of the robotic arm, thus on the angles θ_i and on the geometry, i.e., the length of the joints. Hence the model (3.8) can be translated into the compact measurement equation

$$\mathbf{p}_\text{s}(k) = g_\text{s}(\mathbf{x}(k)) + \mu_\text{s}(k), \tag{3.9}$$

where g_s is the nonlinear function that represents the composition of the three maps in the sensor model (3.8). We add the independent measurements of the angles,

$$\tilde{\theta}_i(k) = \theta_i(k) + \mu_{\theta_i}(k), \quad i = 0, \ldots, J \tag{3.10}$$

(with μ_{θ_i} sensor noise). Denote the measurement vector with $\mathbf{z} = (\mathbf{p}_\text{s}^\top, \tilde{\theta}_0, \ldots, \tilde{\theta}_J)^\top$ and the measurement noise vector with $\boldsymbol{\mu} = (\mu_\text{s}^\top, \mu_{\theta_0}, \ldots, \mu_{\theta_J})^\top$. We can write the complete measurement equation for the robotic arm setup, as:

$$\mathbf{z}(k) = g(\mathbf{x}(k)) + \boldsymbol{\mu}(k). \tag{3.11}$$

We note that (3.11) the stacked representation of (2.3), from which we derive the a posteriori distribution $p(\mathbf{z}(k)|\mathbf{x}(k))$.

In the next subsections we will analyze several experimental and simulation results. The first aim of the experiments is to show the performance of our proposed Algorithm 3.1 in estimating the state $\mathbf{x}$ given the noisy observation $\mathbf{z}$. In particular we will focus only on a part of the state: the position of the object in the world coordinates, i.e., (y, z). We will describe the dependences of the estimation error on the different parameters of the algorithm, as well as its runtime performances. Furthermore, we will show its scalability with respect to the dimension of the state vector (arbitrarily varied in the simulation runs by changing the number of joints J).

The second aim of the experiments is to demonstrate that fast real-time feedback control based on the proposed algorithm is possible and can achieve satisfactory results, in contrast with traditional single-core CPU implementations. The control objective is to track a moving object with the robotic arm while it traverses the screen, as described next.

For experimental and simulation purposes we use the commercially available GTX 580 GPU and, *unless differently stated*, we choose

- a ring topology for the underlying communication graph,
- and we select $t = 1$ for the number of exchanged particles among the computing units.

3.3.2 Experimental Results

We use the robotic arm platform of Figure 3.3 for the experiments throughout this section in order to examine the filter behavior under different conditions. The parameters of the platform and the Particle Filter are listed in Table 3.2 (Experiments column), where all noise terms are modeled as Gaussian (since this turns out to be a rather realistic model for the noise of the setup).

In the experiments the robotic arm is ask to follow a moving object (a white dot) while it traverses the monitor. The robotic arm pose is controlled so that the camera keeps the object in view while staying at a couple of centimeters from the monitor itself (thus the camera does not have a view of the whole monitor), see Table 3.2. The implemented controller is a discrete-time PID controller based on the estimated position and the estimated joint angles with sampling rate of 100 Hz. This update rate is close to the hardware limit.

Table 3.2: *Experiment and Simulation Parameters*

	Experiments	High-noise Simulations
Process Noise*		
$\mathbf{w}_{\theta_i}$	0.015 rad	0.075 rad
$\mathbf{w}_y, \mathbf{w}_z$	0.001 m	0.005 m
$\mathbf{w}_{v_y}, \mathbf{w}_{v_z}$	0.05 m/s	0.25 m/s
Measurement Noise*		
μ_s	10 px	10 px
μ_{θ_i}	0.01 rad	0.01 rad
Other Parameters		
Δt	0.01s	0.01s
Velocity of the target	$\simeq 0.03$ m/s	$\simeq 0.03$ m/s
$\frac{\text{Camera view area}}{\text{Total area}}$	12%	–

* both process noise and measurement noise are chosen to be Gaussian with zero mean and indicated standard deviation.

The performance index we are interested in this subsection is the estimation error of the position of the target. In particular we define the average error, $\mathbf{e}$, as

$$\mathbf{e} = \frac{1}{T_\mathrm{f}} \sum_{k=1}^{T_\mathrm{f}} ||\hat{\mathbf{p}}_\mathrm{w}(k) - \mathbf{p}_\mathrm{w}(k)||, \tag{3.12}$$

where $\hat{\mathbf{p}}_\mathrm{w}(k)$ is the estimated position by the filter and $\mathbf{p}_\mathrm{w}(k)$ is the true position, while T_f is the final discrete time step of the experiment and $||\cdot||$ represents the 2-norm. We remark that by definition $\mathbf{e}$ is always positive, i.e., $\mathbf{e} > 0$.

Figure 3.5 illustrates the estimation of the position (y, z) superimposed on the actual trajectory of the object in low particle and high particle settings. In Figure 3.5.a we have selected $N = 32$ and $m = 64$ and we note that the robotic arm loses track of the object and it cannot complete the trajectory. On the contrary, in Figure 3.5.b, in the setting $N = 2048$ and $m = 512$, we can achieve better estimation which translates in the accomplishment of the object following task. In Figure 3.5.a we have also indicated the dimensions of the camera view area (empty rectangle).

Figure 3.4 shows the average estimation error $\mathbf{e}$ for different settings. We note the general (expected) trend that a higher number of filters increases the accuracy. Moreover, we achieve an average error of 3 mm for the best setting (with 1M particles at 100 Hz) which is considered a remarkable result given the experimental setup.

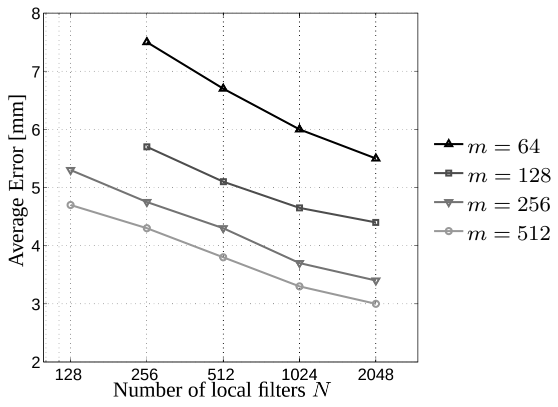
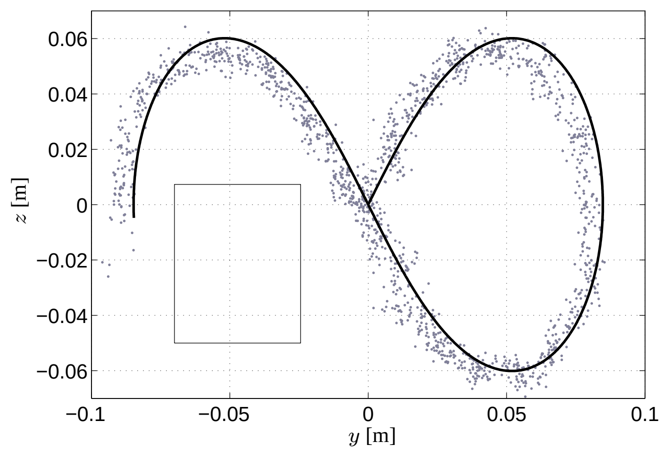


Figure 3.4: *Averaged estimation error* $\mathbf{e}$ *in a number of experiments with different* (N, m) *settings. The standard deviation, not depicted, is around* 1 *mm.*

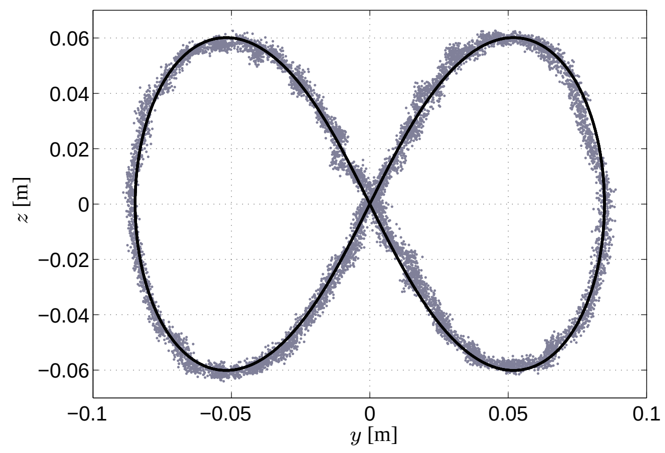
3.3.3 High-noise and Large-scale Simulation Results

In order to further assess our implementation in different scenarios, we simulate the filtering problem in high process noise and large-scale settings. Both the cases serve to illustrate the performance of the algorithm in situations that might be encountered in real-life applications.

First of all we increase the noise parameters as expressed in Table 3.2 and we perform 100 simulation runs for each (N, m) setting. Figures 3.6-3.7 show the results for different topology choices $\mathcal{G}$ and for a different number of exchanged particles t. Although we have performed simulations with several different $\mathcal{G}$ and t we report here only the most indicative ones. As we may note from Figure 3.6 (where we use $t = 1$), in this high-noise setting, the ring topology performs in general better than the all-to-all topology. This is in contrast with the design choice of available algorithms, e.g., (Bashi et al., 2003), where only all-to-all communication is considered. We remark that this effect is due to the lack of diversity in the resampling stage.



(a) $N = 32$ and $m = 64$



(b) $N = 2048$ and $m = 512$

Figure 3.5: *Estimated position (y, z) in grey dots, superimposed on the actual trajectory of the object. The setting $N = 32$ and $m = 64$ does not allow the robotic arm to follow the object, whereas the setting $N = 2048$ and $m = 512$ allows the robotic arm to follow the object very well. Both the experiments have been run at 100 Hz. The empty rectangle on the figure (a) represents the camera view area.*

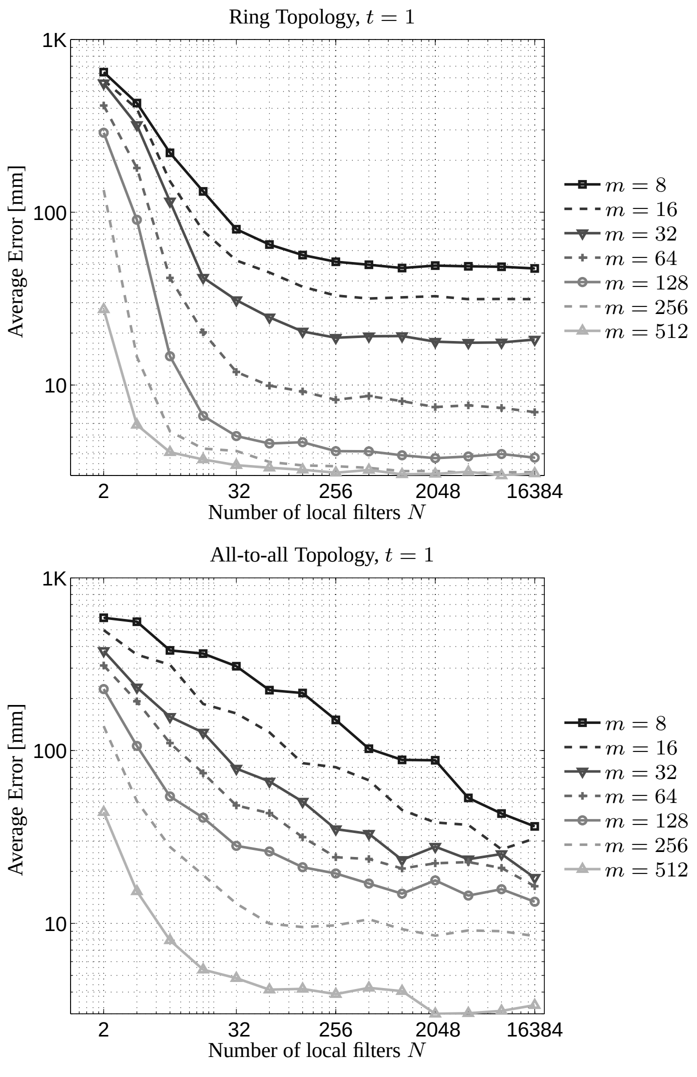
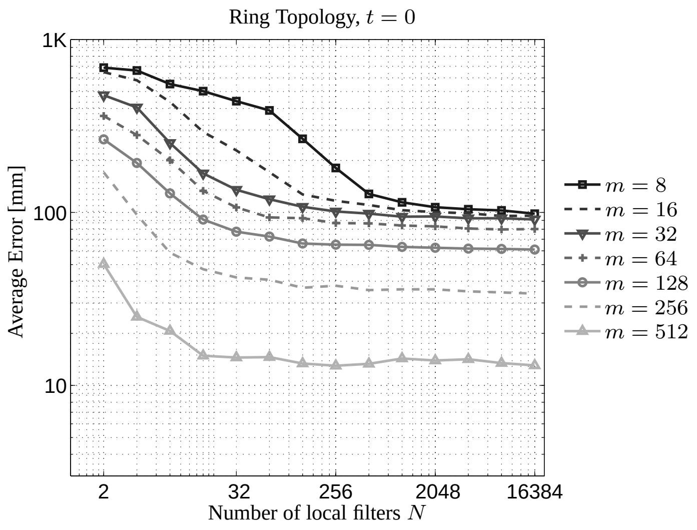


Figure 3.6: *Average error for different (N, m) settings varying the communication topology $\mathcal{G}$.*



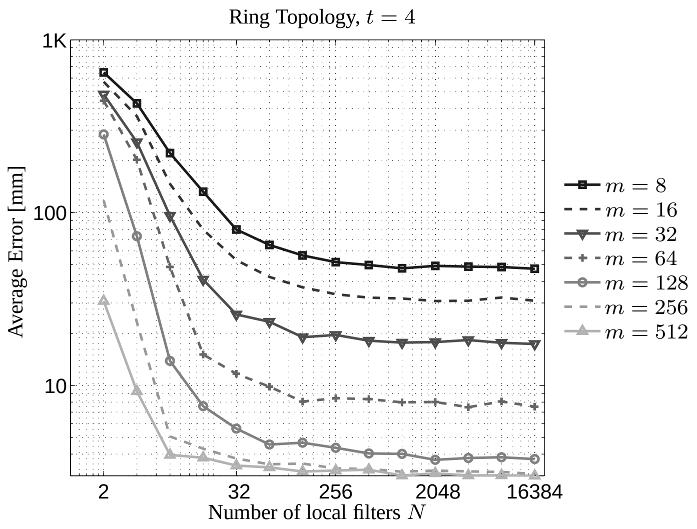


Figure 3.7: *Average error for different (N, m) settings varying the number of shared particles t.*

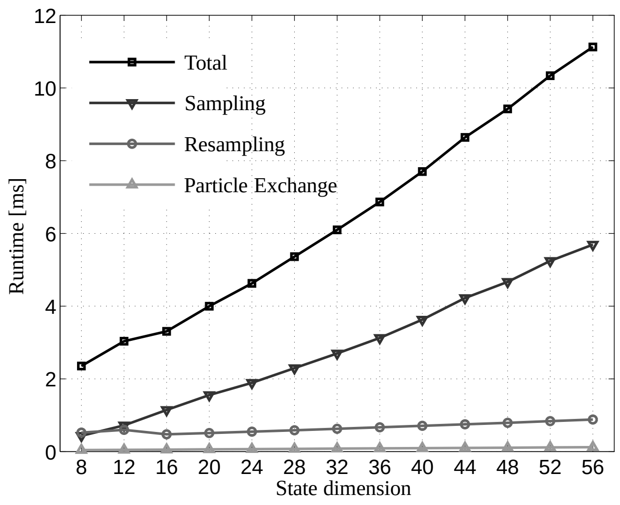


Figure 3.8: *Runtime of the some of the kernels of the distributed computation Particle Filter implementation varying the state dimension in the setting* $N = 2048$ *and* $m = 512$.

Furthermore, from Figure 3.7 (where only the ring topology is used) and Figure 3.6 top we see that even a small number of exchanged particles can make a significant difference compared to the no communication choice ($t = 0$). Moreover, as expected, we note that this difference is not so marked when passing from $t = 1$ to $t = 4$, meaning that the real improvement is in communicating itself and not in the number of exchanged particles. This is in contrast with the design choice of some of the available methods, e.g. (Brun et al., 2002, Bolić et al., 2005, Chao et al., 2010), where either no communication is chosen or 25% of the total particles are shared.

As a second variation on the experimental results, we increase the number of state dimensions augmenting the number of joints J with the setting $N = 2048$, $m = 512$. This case illustrates the scalability of the algorithm, in terms of runtime, with respect to the state dimension. As we see from Figure 3.8 the Particle Exchange step, as well as Resampling, require a relatively limited runtime and they scale better than the sampling step. This was to be expected, since when the state dimension increases, the sampling is the most affected task.[2]

Finally, an important observation from Figure 3.8 is the total absence of bottlenecks in the access to global memory in the Particle Exchange step.

3.3.4 Comparison with a Centralized Sequential Implementation

As a final set of simulation runs, we use the high-noise settings (ref. Table 3.2) and we compare the proposed distributed Particle Filter implementation with a sequential central-

[2]The other kernels are pseudo-random number generation, sorting, local and global estimation.

ized implementation (i.e, a standard Particle Filter that runs sequentially on a single-core architecture).

First of all, we consider in Figure 3.9 the runtime in [ms] for the distributed implementation and the centralized one (the centralized has been implemented on a Intel Core i7-2820QM processor running at 2.3 GHz). We set $m = 512$ and we vary N. As we note from Figure 3.9, the centralized algorithm scales exponentially in the number of particles (as we should have expected). The proposed distributed method instead scales better (increasing N) and for an high particle setting ($Nm >$ 16K), it is from 10 to more than 100 times faster than the sequential centralized implementation. The contrary, meaning the distributed particle filter is slower than the centralized implementation, is instead reasonable with a low number of particles ($Nm <$ 1K), since the single core is more powerful computationally-wise than the local cores of the GPU architecture and there is no communication involved.

Figure 3.9 can be also used for a comparison with the methods presented in the literature. With reference to Table 3.1, we can report in Table 3.3 the performance of the proposed method.

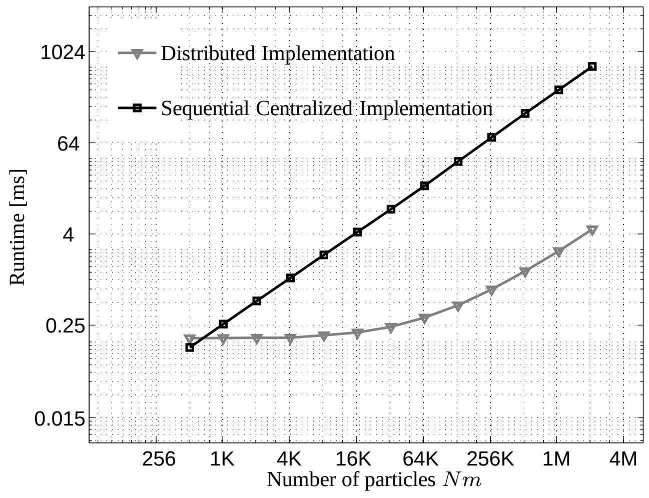


Figure 3.9: *Runtime comparison of the proposed distributed implementation and a sequential centralized implementation. In the distributed one we fix $m = 512$ and we vary N.*

Table 3.3: *Performance of the proposed approach*

Ref.s	Sampling + weight	Resampling	Estimation	Particles	State dimension	Runtime [ms]
Algorithm 3.1	local	distributed	local	64K	8	0.3
				1M	8	2.3
				4M	8	4.6

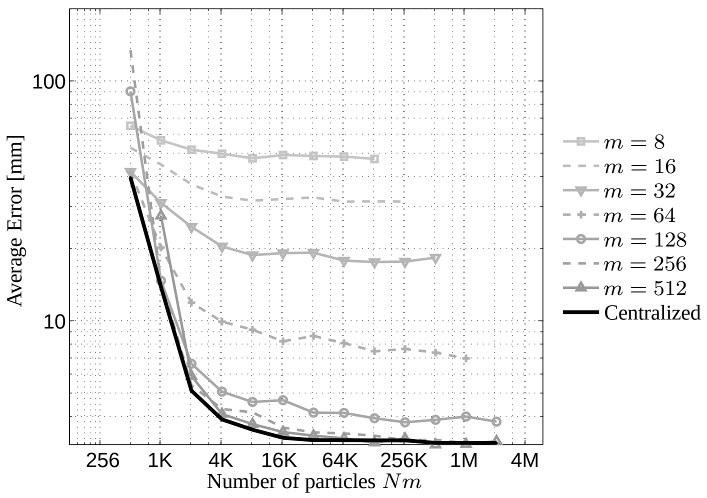


Figure 3.10: *Comparison of the estimation error between distributed implementation and centralized one.*

As we see from Table 3.3 the presented method outperforms for number of particles, state dimension, and/or runtime state-of-the-art methods employing GPU architectures. Furthermore, Algorithm 3.1 increases the achievable performances often by orders of magnitude.

In Figure 3.10 we report the estimation error in both the distributed implementation and the centralized one. As we notice, the estimation error for cases in which $m \geq 128$ is comparable with the centralized setting. Furthermore, in the case of $m = 512$ and $N \geq 1024$, the distributed algorithm delivers better estimates than the centralized one. This has to do with the loss of diversity in the centralized implementation.

Figure 3.10 gives also extra insights on the selection of m and N given a total number of particles Nm. In fact, for high value of Nm it appears better, in terms of estimation error, to choose a high value for m. This configuration leads to a small number of accurate filters. For low Nm settings, the opposite seems to be more recommended. This lead to an high number of less accurate filters (but yet with more particle diversity).

3.4 Conclusions

In this chapter we have shown that fast yet accurate nonlinear estimation is realizable and it can be used in relatively high sample rate real-time feedback controller. In particular we have designed, analyzed, and implemented a distributed computation Particle Filter that can handle over a million particles at 100 Hz with remarkable estimation accuracy. This result outperforms other implementations that can be found in literature. In particular, our implementation increases the number of particles, the state dimension, and/or the sampling frequency often by orders of magnitude with respect to state-of-the-art GPU solutions (typically based on parallel algorithms instead of our distributed ones).

Furthermore, we have shown that the proposed scheme has comparable accuracy with centralized sequential particle filters (with the same number of total particles), which require 10-100 times more computational time (when using a high number of particles) than our proposed distributed implementation.

3.5 Open Problems and Future Work

As future work directions we can recommend the following two open problems.

Optimal selection of the couple (N, m)

The first open problem is the optimal selection of the number of local filters N and the number of local particles m to guarantee a certain level of accuracy and sampling rate. This selection would help the users to choose the right couple (N, m) for their own application.

Although this is not expected to be an easy problem, we believe that even the intermediate step of having some good rule of thumb would already be beneficial in order to guide the user in the right direction.

Distributed Sensing and Computation Algorithms

The second open research question is the design of algorithms that merge distributed sensing estimators of Chapter 2 with distributed computation ones of this chapter, to obtain a fast and accurate nonlinear estimator that can work well also in a distributed sensing scenario.

Chapter 4

Distributed Control of Robotic Networks with State-Dependent Laplacians

***Abstract* —** This chapter considers two distributed control problems for robotic networks.

First, we analyze the problem of maximizing the algebraic connectivity of the communication graph in a network of mobile robots by moving them into appropriate positions. We define the Laplacian of the graph as dependent on the pairwise distance between the robots and we approximate the problem as a sequence of Semi-Definite Programs (SDP). We propose a distributed solution consisting of local SDP's which use information only from nearby neighboring robots. We show that the resulting distributed optimization framework leads to feasible subproblems and through its repeated execution, the algebraic connectivity increases monotonically. Moreover, we describe how to adjust the communication load of the robots based on locally computable measures.

Second, we utilize and extend the presented distributed method to tackle the problem of collectively tracking a number of moving targets while maintaining a certain level of connectivity among the network of mobile robots. We formulate the combined global objective also as a Semi-Definite Program (SDP) and propose a non-iterative distributed solution consisting of localized SDP's which use information only from nearby neighboring robots.

Numerical simulations show the performance of the distributed algorithms with respect to the centralized solutions.

4.1 Introduction

In Chapter 2 we have analyzed situations in which the computing and communicating devices were non-moving. In this chapter we shift our focus on mobile devices, such as mobile robots, that have to be controlled to achieve a common task.

These teams of autonomous mobile robots are considered as a key enabling technology in several applications ranging from underwater and space exploration (Leonard et al., 2010, Izzo and Pettazzi, 2007), to search, rescue, disaster relief (Lau and Ko, 2008, Casper and Murphy, 2003), and monitoring and surveillance (Casbeer et al., 2005, Mathews and Durrant-Whyte, 2007). Among the engineering and research questions these applications pose, maintaining a good level of connectivity between the individual robots and increasing the communication quality given the environmental constraints, have fundamental importance. Nonetheless, as expressed in Chapter 1, this aspect has been often overlooked in the available literature.

In this chapter, we will use the algebraic connectivity, or $\lambda_2(\mathcal{G})$, of the communication graph as a natural measure of connectivity "quality", and we recall that a strictly positive $\lambda_2(\mathcal{G}) > 0$ is a necessary and sufficient condition for the graph $\mathcal{G}$ to be connected.

In the context, in Section 4.2, we study distributed solutions for maximizing $\lambda_2(\mathcal{G})$ in mobile robotic networks. We will focus on distance-based connectivity maximization with minimum separation constraints, as opposed to ensuring line-of-sight connectivity in an obstacle-rich environment (Anisi et al., 2008). To the best of our knowledge, only the work of (De Gennaro and Jadbabaie, 2006) investigates a distributed solution for the maximization of λ_2 based on a simplified scenario where the dynamics of the robots are represented by a single integrator and no constraints are present. The authors use a two-step distributed algorithm, which relies on super-gradients and potential functions. The required communication load scales with the square of the graph diameter which may impede fast real-time implementations for large groups of robots.

In Section 4.2 we consider as starting point the centralized optimization procedure of (Kim and Mesbahi, 2006, Boyd, 2006, Derenick et al., 2009). In these works the maximization of the algebraic connectivity is approximated as a sequence of Semi-Definite Programs based on the notion of state-dependent graph Laplacian, while the agents are modeled as discrete-time single integrators.

First, we modify the aforementioned centralized optimization procedure in order to handle more realistic robot dynamics. The resulting optimization problem is then proven to be feasible at each time step under quite general assumptions. Second, we propose a distributed solution for the modified centralized problem. Our proposed distributed approach relies on local problems that are solved by each robot using information only from nearby neighbors and, in contrast with (De Gennaro and Jadbabaie, 2006), it does not require any iterative schemes, making it more suitable for real-time applications. In our approach *(i)* we formulate local problems of small size that are clearly related to the centralized one, *(ii)* the *linearized* algebraic connectivity of the approximate problem is guaranteed to be monotonically increasing, *(iii)* the overall optimization scheme is proven to be feasible at each time step under quite general assumptions, and in particular *(iv)* the local solutions are feasible with respect to the constraints of the original centralized problem. Finally, we characterize the local relative sub-optimality of the optimized algebraic connectivity with respect to a larger neighborhood size and we use this characterization to enable each robot to increase or decrease its communication load on-line, while respecting the properties *(i)* - *(iv)*. This means that our solution can be adapted based on available resources, augmenting or reducing the required communication and computational effort.

Besides its benefits in improving communication quality, the proposed distributed solution can be adapted to certain situations in which the connectivity level is a constraint while the robots are performing other tasks. An example of these scenarios is multi-target tracking, which is the topic of Section 4.3.

In Section 4.3, we start formulating the multi-target tracking problem as a generalization of the approach of (Derenick et al., 2009, 2010). This generalization accounts for uncertainty in the targets' positions, which is relevant in tracking scenarios. Then, we formulate an optimization problem as a joint maximization of connectivity among the robots and visibility of the moving targets and we extend the distributed solution proposed in Section 4.2 to this problem formulation. The distributed solution, in addition to feasibility with respect to the constraints and to the non-iterative nature, can guarantee both connectivity and tracking, and the local cost functions exhibit the same improvement property as the global cost in a linearized approximation.

The distributed solutions of this chapter can be seen as a complementary approach to standard subgradient algorithms. Distributed versions of incremental subgradient algorithms are typically communication intensive iterative algorithms, in which at each iteration, each agent has to evaluate only a subgradient of a certain function. Our proposed solutions lie on the other side of the "communication-computation" trade-off spectrum. In fact, each robot solves a reasonably complex convex optimization problem, while the communication among them remains limited.

4.2 Constrained Algebraic Connectivity Maximization

4.2.1 Problem Formulation

Consider a network of N agents with communication and computation capabilities and express as $a_i(k)$ the value of the variable a for agent i at the discrete time instant k. These agents can be thought of as a representation of the aforementioned mobile robots. The position of agent i is denoted by $x_i(k) \in \mathbb{R}^3$ and its velocity by $v_i(k) \in \mathbb{R}^3$. To begin with, assume the agents to move according to the following discrete-time dynamical system

$$x_i(k+1) = x_i(k) + v_i(k)\Delta t, \tag{4.1}$$

where Δt is the sampling time. This single-integrator model (4.1) will be used to introduce the works of (Kim and Mesbahi, 2006, Boyd, 2006, Derenick et al., 2009) which enable us to elaborate our contributions in subsequent sections, where we consider more complex agent dynamics.

Graph-theoretic notions are used to model the network. Let $x(k)$ be the stacked vector containing the positions of the agents, i.e. $x(k) = (x_1^\top(k), \ldots, x_N^\top(k))^\top$. The set $\mathcal{V}$ contains the indices of the mobile agents (nodes), with cardinality $N = |\mathcal{V}|$. The set $\mathcal{E}$ indicates the set of communication links. The graph $\mathcal{G}$ is then expressed as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and it is assumed undirected. Let the agent clocks be synchronized, and assume perfect communication (no delays or packet losses). The agents with which agent i communicates are called neighbors and are contained in the set $\mathcal{N}_i$. Note that agent i is not included in the set $\mathcal{N}_i$. We define $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{i\}$ and $N_i = |\mathcal{N}_i^+|$. Define the Laplacian matrix L associated with $\mathcal{G}$ via its entries $[L]_{ij}$ as

$$[L]_{ij}(k) = \begin{cases} 0 & (i,j) \notin \mathcal{E} \\ -w_{ij}(k) & (i,j) \in \mathcal{E}, i \neq j \\ \sum_{l \neq i} w_{il}(k) & i = j \end{cases} \tag{4.2}$$

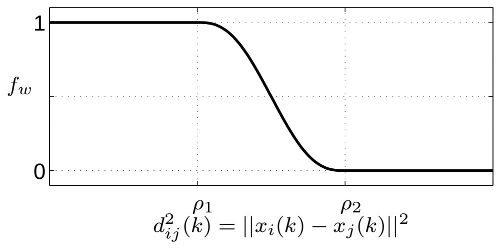


Figure 4.1: *Weighting function $f_w(\cdot)$ for modeling connectivity between two agents i, j. If $d_{ij}^2(k) < \rho_1$ then $w_{ij} = 1$, while if $d_{ij}^2(k) > \rho_2$ then $w_{ij} = 0$.*

The weights $0 \leq w_{ij}(k) \leq 1$ are assumed to depend on the squared Euclidean distance of $x_i(k)$ and $x_j(k)$ defined as

$$d_{ij}^2(k) = f_d(x_i(k), x_j(k)) = ||x_i(k) - x_j(k)||^2 \tag{4.3}$$

and

$$w_{ij}(k) = f_w(||x_i(k) - x_j(k)||^2) \tag{4.4}$$

where $f_w : \mathbb{R}_+ \to [0, 1]$ is a smooth nonlinear function with compact support.[1] The weights model the connection strength between two agents. The closer two agents are, the closer to one is the weight, representing an increase in the communication "quality". For simulation purposes we use the function qualitatively represented in Figure 4.1, which is one when the squared distance is less than ρ_1 and it is zero when the squared distance is greater than ρ_2. For a detailed discussion on the choice of f_w the reader is referred to (Kim and Mesbahi, 2006). As a direct consequence of the above definitions, the entries of the Laplacian matrix L depend on the state of the agents, making it state-dependent, which we will denote by $L(x(k))$. By construction, the Laplacian is a positive semidefinite matrix, with real eigenvalues ordered in a crescent way as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. The smallest eigenvalue is always 0 and its associated eigenvector is $\mathbf{1}_N$. The second smallest eigenvalue of the Laplacian is often referred to as the algebraic connectivity of the graph and indicated as $\lambda_2(\mathcal{G})$, or $\lambda_2(L)$ (In the following, we will write $\lambda_2(x(k))$ denoting that also the algebraic connectivity depends on the state). The algebraic connectivity is a "measure" of connectivity since

- a zero value for the algebraic connectivity, i.e., $\lambda_2(L) = 0$, implies that the graph is not connected;
- if $\lambda_2(L) > \lambda_2(L')$ then L has more links than L', or the links have more weight (loosely speaking, L is better connected than L').

For the reason of increasing the connectivity of the communication graph among the moving agents, we are interested in maximizing the algebraic connectivity. We will achieve

[1]Functions with compact support in $\mathbb{R}_+$ are those with support that is a compact subset of $\mathbb{R}_+$.

this by by controlling the state of the agents, i.e., moving them to appropriate positions. This goal is formulated as the following time-invariant optimization problem:

Problem 4.1 Algebraic Connectivity Maximization

$$\mathbf{P}\left(\lambda_2(x), \rho_1\right): \quad \underset{x}{\mathbf{maximize}} \quad \lambda_2(x) \tag{4.5a}$$

$$\mathbf{subject\ to} \quad f_d(x_i, x_j) > \rho_1, \quad \forall (i,j) \in \mathcal{E} \tag{4.5b}$$

The optimal decision variables are the final robot locations x. Here the constraint on $f_d(x_i, x_j)$ prevents the agents from getting too close to each other and ensures that the trivial solution in which all the agents converge to one point is not part of the feasible solution set of (4.8a).

4.2.2 Centralized Approach

First of all, we rewrite the problem (4.5a) in terms of matrix inequalities. We use the following lemma.

Lemma 4.1 *For any two scalars $\lambda > \bar{\lambda}_2 > 0$, the constraint*

$$\lambda_2(L) > \bar{\lambda}_2, \tag{4.6}$$

can be formulated with the equivalent Matrix Inequality

$$L + (\lambda/N)\mathbf{1}_N\mathbf{1}_N^\top \succ \bar{\lambda}_2 I_N. \tag{4.7}$$

Proof. By construction, the Laplacian matrix L has as eigenvector $\mathbf{e}_1 = \mathbf{1}_N$. All the other eigenvectors, $\mathbf{e}_i$, are orthogonal to $\mathbf{1}_N$, meaning $\mathbf{1}_N^\top\mathbf{e}_i = 0$, for $i = 2, \ldots, N$. This implies that

$$\left(L + (\lambda/N)\mathbf{1}_N\mathbf{1}_N^\top\right)\mathbf{e}_i = L\mathbf{e}_i = \lambda_i\mathbf{e}_i, \quad \text{for } i = 2, \ldots, N$$

and therefore $L + (\lambda/N)\mathbf{1}_N\mathbf{1}_N^\top$ has the same eigenvalues/eigenvectors of L for $i = 2, \ldots, N$. The remaining eigenvalue is associated with the $\mathbf{e}_1$ eigenvector:

$$\left(L + (\lambda/N)\mathbf{1}_N\mathbf{1}_N^\top\right)\mathbf{e}_1 = L\mathbf{1}_N + (\lambda)\mathbf{1}_N = \lambda\mathbf{1}_N$$

and its value is λ. As a result, the eigenvalues of $L + (\lambda/N)\mathbf{1}_N\mathbf{1}_N^\top$ are

$$\lambda, \lambda_2(L), \lambda_3(L), \ldots, \lambda_N(L).$$

Since we have already that $\lambda > \bar{\lambda}_2$ (by assumption), and $\lambda_2(L) \leq \lambda_3(L) \leq \ldots \lambda_N(L)$, the constraint (4.7) imposes that $\lambda_2(L) > \bar{\lambda}_2$ and thus it is equivalent to (4.6). $\square$

Since for the specified weighted Laplacian $L(x)$ the maximum value for λ_2 is $N - 1$ (de Abreu, 2007), we can chose $\lambda = N$ in (4.7) and rewrite problem (4.5a) in the equivalent formulation:

$$
\begin{aligned}
\mathbf{P}\left(L(x),\rho_1\right): \quad & \underset{x,\gamma}{\textbf{maximize}} \quad && \gamma && (4.8a)\\
& \textbf{subject to} && \gamma \geq 0 && (4.8b)\\
& && L(x) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma I_N && (4.8c)\\
& && f_d(x_i,x_j) > \rho_1, \quad \forall (i,j)\in\mathcal{E} && (4.8d)
\end{aligned}
$$

Problem (4.8a) is non-convex but it is rather standard to obtain a time-varying convex approximation by using first-order Taylor expansions, (Kim and Mesbahi, 2006, Derenick et al., 2009). Define

$$c_{ij}^w = \left.\frac{\partial f_w}{\partial d_{ij}^2}\frac{\partial d_{ij}^2}{\partial x_i}\right|_{x_i(k),x_j(k)} = -\left.\frac{\partial f_w}{\partial d_{ij}^2}\frac{\partial d_{ij}^2}{\partial x_j}\right|_{x_i(k),x_j(k)}, \tag{4.9}$$

$$c_{ij}^d = \left.\frac{\partial f_d}{\partial x_i}\right|_{x_i(k),x_j(k)} = -\left.\frac{\partial f_d}{\partial x_j}\right|_{x_i(k),x_j(k)} \tag{4.10}$$

then we can write the approximations

$$w_{ij}(k+1) = w_{ij}(k) + {c_{ij}^w}^\top(\delta x_i(k+1) - \delta x_j(k+1)) \tag{4.11}$$

$$d_{ij}^2(k+1) = d_{ij}^2(k) + {c_{ij}^d}^\top(\delta x_i(k+1) - \delta x_j(k+1)) \tag{4.12}$$

where δ represents the difference operator, i.e. $\delta x_i(k+1) = x_i(k+1) - x_i(k)$. The symbol $\triangle$ will be employed to define the linearized entities; hence the entry $[\triangle L]_{ij}(x(k+1))$ of the Laplacian $\triangle L(x(k+1))$ will be

$$[\triangle L]_{ij}(x(k+1)) = \tag{4.13}$$

$$\begin{cases} 0 & (i,j)\notin\mathcal{E}\\ -w_{ij}(k) - {c_{ij}^w}^\top(\delta x_i(k+1) - \delta x_j(k+1)) & (i,j)\in\mathcal{E}, i\neq j\\ \sum_{l\neq i} w_{il}(k+1) & i=j \end{cases}$$

while

$$\triangle f_d(x_i(k+1), x_j(k+1)) = d_{ij}^2(k) + {c_{ij}^d}^\top(\delta x_i(k+1) - \delta x_j(k+1)) \tag{4.14}$$

This allows us to consider the maximization of the algebraic connectivity of L as the following time-varying convex optimization problem (Kim and Mesbahi, 2006, Boyd, 2006, Derenick et al., 2009):

$$\triangle\mathbf{P}\left(L(x(k)), x(k), \mathcal{S}_{\triangle\mathcal{Q}_2}\right): \underset{x(k+1),\gamma(k+1)}{\textbf{maximize}}\ \gamma(k+1) \tag{4.15a}$$

$$\textbf{subject to}$$

$$\triangle\mathcal{Q}_1: \begin{cases} \gamma(k+1) \geq 0 \\ \triangle L(\mathbf{x}(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma(k+1)I_N \end{cases} \tag{4.15b}$$

$$\triangle\mathcal{Q}_2: \begin{cases} \mathcal{Q}_{2.1}: & \triangle f_d(x_i(k+1), x_j(k+1)) > \rho_1, \\ & \forall(i,j) \in \mathcal{E} \\ \mathcal{Q}_{2.2}: & ||x_i(k+1) - x_i(k)|| \leq v_{\max}\Delta t \\ & i = 1, \ldots, N \end{cases} \tag{4.15c}$$

where $\mathcal{S}_{\triangle\mathcal{Q}_2} = \{\rho_1, v_{\max}\}$ represents the parameter set that characterizes the set of constraints $\triangle\mathcal{Q}_2$, and it is used to highlight the dependence of the problem on the "physical" limitation of the application scenario (i.e., in this case, the mutual distance ρ_1 and the maximum allowed velocity $v_{\max}$). It will be shown later how this parameter set will change in the different formulations of the problem.

In contrast to the original non-convex problem (4.8a), the optimization problem (4.15a) is solved *repeatedly* at each discrete time step k on-line. In this sense (4.15a) is the k-th problem of a sequence of convex SDP problems, and therefore this approach could be regarded as sequential convex programming. Note that the achieved maximal algebraic connectivity γ depends on k and thus we use $\gamma(k)$. In this sense the iterative scheme for updating γ is the repeated solution of the optimization problem itself. We remark that as a consequence of using this sequence of convex programs (and as a consequence of the non-convex nature of the original problem), although we aim at increasing the cost function at each step k, we might converge to a local minimum of the original problem (4.8a) and a strong dependence on the initial configuration of the agents is to be expected. Despite these drawbacks, convergence has been proven in (Kim and Mesbahi, 2006), where the authors have also shown that this formulation does indeed lead to satisfactory local optimal final configurations with a clear increase in the algebraic connectivity.

Remark 4.1 *The reader is referred to Section 4.5 for further considerations on the adopted linearization procedure and its possible improvements.*

If we assume that the initial positions $x(0)$ form a connected graph and the mutual distance between the agents is greater than $\sqrt{\rho_1}$, i.e., we assume initial feasibility for the problem, we can easily prove that the optimization problems (4.15a) will remain feasible for all the subsequent time steps $k > 0$ (in fact one can always select $x(k) = x(k+1)$ to obtain a feasible solution) and their solution sequence monotonically increases the algebraic connectivity, (Kim and Mesbahi, 2006). The property of remaining feasible for all k is related to *persistent* feasibility (also known as *recursive* feasibility), which is a well-known and fundamental concept in the optimization-based control literature (Borrelli et al., 2011). In particular, persistent feasibility ensures that, for any k, if the k-th convex problem (4.15a) is feasible then the $(k+1)$-st problem will be feasible. This, in addition to initial feasibility (i.e., feasibility at $k = 0$), guarantees that the overall sequential optimization scheme is feasible for all $k > 0$. It has to be noted that persistent feasibility ensures only that the solution set of each problem (4.15a) is non-empty, while any improvement in the cost function should be proven separately. However, persistent feasibility is needed in the first place to justify the overall optimization scheme in practice.

4.2.3 Extension of the Centralized Approach

As our first contribution of the chapter, we extend the problem (4.15a) in order to allow a more realistic dynamical model for the agents. In the following, the state of the agents is augmented to include not only their position but also their velocity. Let $\mathbf{x}_i(\tau) = (x_i(\tau)^\top, v_i(\tau)^\top)^\top$ be the state of agent i at the discrete time τ. We note that the sampling periods belonging to τ and k may differ, meaning that the optimization (4.15a) could be run at a slower rate than the system dynamics. Let the agents have the following second order discrete-time LTI dynamics:

$$\begin{pmatrix} x_i(\tau+1) \\ v_i(\tau+1) \end{pmatrix} = \begin{pmatrix} I_3 & A_{1i} \\ 0_3 & A_{2i} \end{pmatrix} \begin{pmatrix} x_i(\tau) \\ v_i(\tau) \end{pmatrix} + \begin{pmatrix} 0_3 \\ b_{1i} I_3 \end{pmatrix} u_i(\tau) \tag{4.16}$$

where $A_{1i} \in \mathbb{R}^{3\times3}$, $A_{2i} \in \mathbb{R}^{3\times3}$, $b_{1i} \in \mathbb{R}_0$, and $u_i(\tau) \in \mathbb{R}^3$ is the control input. Assume:

Assumption 4.1 *The matrix A_{1i} is full rank $\forall i$.*

Assumption 4.2 *The control input for each agent at each discrete time step is constrained in the closed polytopic set $\bar{\mathcal{U}}_i$:*

$$u_i(\tau) \in \bar{\mathcal{U}}_i,\, \bar{\mathcal{U}}_i = \{u_i(\tau) \in \mathbb{R}^3 | H_i u_i(\tau) \leq h_i\}, \mathbf{0}_3 \in \bar{\mathcal{U}}_i \tag{4.17}$$

described via the matrix H_i and the vector h_i.

Assumption 4.1 is meant to ensure the one-step controllability of the dynamical system described in Eq. (4.18). Analogously to $v_{\max}$ in problem (4.15a), Assumption 4.2 limits the control input to account for the physical limitations of the agents, and it is a standard formulation of actuator limitations in the optimization-based control community.

The state space system in (4.16) can model agents for which the acceleration does not depend on the position and for which zero velocity and acceleration input ($v_i(\tau) = 0$ and $u_i(\tau) = 0$) implies $x_i(\tau+1) = x_i(\tau)$. Typically, this class of systems can represent different types of physical agents ranging from fully actuated mobile robots to underwater vehicles (see Remark 4.2). The choice $A_{1i} = I_3\Delta t$, $A_{2i} = I_3$, $b_{1i} = \Delta t$ yields a double integrator with sampling period Δt. The reason for the choice of (4.16) is to consider the simplest model that is capable of showing how to handle the main difficulties when extending the optimization problem (4.15a) to general LTI models. In particular, the key issues are persistent feasibility and collision avoidance. In order to guarantee persistent feasibility we show how to ensure that $\mathbf{x}_i(k+1) = (x_i^\top(k), \mathbf{0}_3^\top)^\top$ is a feasible state for all the agents recalling that the feasibility of the similar solution $x_i(k+1) = x_i(k)$ is a sufficient condition for (4.15a) to be persistently feasible. The collision avoidance issue is due to the fact that the constraint on $f_d(x_i(k), x_j(k))$ is enforced only at each time step k, when the optimization problem is solved, but not for every τ, which might be a higher rate implementation of the dynamical model. In this respect we show how to ensure that $f_d(x_i(\tau), x_j(\tau)) > 0$ for every τ. We will show that when persistent feasibility and collision avoidance are handled correctly, the problem (4.15a) can be extended to dynamical models of the form (4.16). In Section 4.5 we discus how to possibly cope with these two aspects for an even broader class of dynamical systems.

Remark 4.2 *Examples of physical agents that can be represented by* (4.16) *are fully actuated (omni-directional) mobile robots or underwater vehicles.*

Consider the first case. Let $x \in \mathbb{R}^2$ be the position and $v \in \mathbb{R}^2$ the velocity. Let $u \in \mathbb{R}^2$ be the velocity control applied at the step τ as

$$v(\tau+1) = v(\tau) + u(\tau).$$

This represents a step in the velocity. The dynamical system of the robot can be modeled as (4.16) *with $A_{1i} = I_2\Delta t$, $A_{2i} = I_2$, $b_{1i} = 1$.*

A similar model could hold also for an underwater vehicle if we consider discrete step in the velocity along three different axes.

Persistent Feasibility

The first step to guarantee persistent feasibility is to ensure that at each time step τ we can affect the position of the agents via the control input. This is not trivial because the position $x_i(\tau+1)$ cannot be controlled in one step by $u_i(\tau)$. However, we can overcome this issue by solving the optimization problem at a slower rate than the implementation of the control input, e.g., every odd τ, when we determine both $u_i(\tau)$ and $u_i(\tau+1)$. In this case the dynamical system (4.16) can be lifted as used in the optimization problem:

$$\begin{pmatrix} x_i(\tau+2) \\ v_i(\tau+2) \end{pmatrix} = \begin{pmatrix} I_3 & A_{1i}(I_3 + A_{2i}) \\ 0_3 & A_{2i}^2 \end{pmatrix} \begin{pmatrix} x_i(\tau) \\ v_i(\tau) \end{pmatrix} + \begin{pmatrix} b_{1i}A_{1i} & 0_3 \\ b_{1i}A_{2i} & b_{1i}I_3 \end{pmatrix} \begin{pmatrix} u_i(\tau) \\ u_i(\tau+1) \end{pmatrix}, \quad (4.18)$$

where, we let $k = \tau/2$, and for integer k's, we define the lifted variables $x_i^L(k) = x_i(\tau)$, $v_i^L(k) = v_i(\tau)$, the lifted state $\mathbf{x}_i^L(k) = (x_i^L(k)^\top, v_i^L(k)^\top)^\top$, and the lifted control input $\mathbf{u}_i^L(k) = (u_i(\tau)^\top, u_i(\tau+1)^\top)^\top$. For the sake of simplicity, from now on, we will omit the superscript L with the idea that if we use the index k we are referring to the lifted variables. With this in mind, we can rewrite the system (4.18) using the short-hand notation

$$\mathbf{x}_i(k+1) = \mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k)) \quad (4.19)$$

We note that the lifted system (4.19) is controllable to an arbitrary state in one step from k to $k+1$. However, the input is constrained to lie in $\mathbf{u}_i(k) \in \mathcal{U}_i$ (Assumption 4.2), where $\mathcal{U}_i = \bar{\mathcal{U}}_i \times \bar{\mathcal{U}}_i$, i.e.:

$$\mathcal{U}_i = \left\{ \mathbf{u}_i(k) \in \mathbb{R}^6 \,\middle|\, \begin{pmatrix} H_i & \\ & H_i \end{pmatrix} \mathbf{u}_i(k) \le \begin{pmatrix} h_i \\ h_i \end{pmatrix} \right\}, \mathbf{0}_6 \in \mathcal{U}_i \quad (4.20)$$

Therefore, the next step is to find a feasible control input value $\mathbf{u}_i(k) \in \mathcal{U}_i$ for which $\mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k)) = (x_i(k)^\top, \mathbf{0}_3^\top)^\top$. For this reason define the set $\mathcal{F}_i$ as

$$\mathcal{F}_i = \{\mathbf{x}_i(k) \in \mathbb{R}^6 \big| \exists \mathbf{u}_i(k) \in \mathcal{U}_i \text{ such that :} \\ \mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k)) = (x_i(k)^\top, \mathbf{0}_3^\top)^\top, \forall k \in \mathbb{N}^+\} \tag{4.21}$$

For the system (4.18) the set $\mathcal{F}_i$ can be computed as the Cartesian product of $\mathcal{F}_{x,i}$ and $\mathcal{F}_{v,i}$, i.e., $\mathcal{F}_i = \mathcal{F}_{x,i} \times \mathcal{F}_{v,i}$, where:

$$\begin{aligned} \mathcal{F}_{x,i} &= \left\{x_i(k) \in \mathbb{R}^3\right\}, \qquad \text{and} \\ \mathcal{F}_{v,i} &= \left\{v_i(k) \in \mathbb{R}^3 \middle| -\begin{pmatrix} H_i b_{1i}^{-1}(I_3 + A_{2i}) \\ H_i b_{1i}^{-1} A_{2i}(I_3 + 2A_{2i}) \end{pmatrix} v_i(k) \leq \begin{pmatrix} h_i \\ h_i \end{pmatrix}\right\} \end{aligned} \tag{4.22}$$

We note that $(x_i(k)^\top, \mathbf{0}_3^\top)^\top \in \mathcal{F}_i$.

Remark 4.3 *The dynamical system (4.18), which is the agent representation seen by the optimization problem, is controllable in one step by an unconstrained $\mathbf{u}_i(k)$. In fact, given an arbitrary state vector $\mathbf{x}_i(k+1)$ and any initial condition $\mathbf{x}_i(k)$, due to the full rank condition on A_{1i} (Assumption 4.1), one can promptly invert the system (4.18) and obtain the (finite) control vector $\mathbf{u}_i(k)$.*

Collision Avoidance

In order to avoid collisions, a lower bound on ρ_1 needs to be determined, which guarantees that, given the distance bound ρ_1:

For each k and for $\tau = 2k+1$, if $f_d(x_i^L(k), x_j^L(k)) > \rho_1$ and $f_d(x_i^L(k+1), x_j^L(k+1)) > \rho_1$ then $f_d(x_i(\tau+1), x_j(\tau+1)) > 0$.

The collision-free condition for any couple i and j can be written as

$$||x_i(\tau+1) - x_j(\tau+1)|| > 0 \tag{4.23}$$

and using the dynamical equation (4.16) we obtain

$$||x_i(\tau) - x_j(\tau) + A_{1i}v_i(\tau) - A_{1j}v_j(\tau)|| > 0. \tag{4.24}$$

We can employ then the triangle inequality to write

$$||x_i(\tau) - x_j(\tau) + A_{1i}v_i(\tau) - A_{1j}v_j(\tau)|| > \\ ||x_i(\tau) - x_j(\tau)|| - ||A_{1i}v_i(\tau) - A_{1j}v_j(\tau)|| > 0. \tag{4.25}$$

Since $||x_i(\tau) - x_j(\tau)|| > \sqrt{\rho_1}$ the worst case scenario can be computed by maximizing the term $||A_{1i}v_i(\tau) - A_{1j}v_j(\tau)||$ over $v_i(\tau) \in \mathcal{F}_{v,i}$ and $v_j(\tau) \in \mathcal{F}_{v,j}$.

This can be rewritten as a non-convex QP problem[2] and solved for any pair i and j. If $\sqrt{\bar{\rho}_1}$ denotes the worst case $||A_{1i}v_i(\tau) - A_{1j}v_j(\tau)||$ over all the pairs, then the collision-free condition (4.25) can be expressed as $\rho_1 > \bar{\rho}_1$. This is a condition that has to be imposed when designing the ρ_1 value in the minimal distance constraint $\mathcal{Q}_{2.1}$. In this respect, we note that the calculations performed to compute $\bar{\rho}_1$ can be made off-line before running the optimization algorithm (and therefore even the non-convex nature of the problem given the small-size and the off-line calculations can be handled in a satisfactory way in practice).

Centralized Problem Formulation

The optimization problem (4.15a) for the maximization of the algebraic connectivity can now be extended for the more general dynamics (4.16) as

$$\triangle\mathbf{P}\left(\triangle L(x), \mathbf{x}(k), \mathcal{S}_{\triangle\mathcal{Q}_2}\right): \quad \underset{\mathbf{x}(k+1),\mathbf{u}(k),\gamma(k+1)}{\textbf{maximize}} \quad \gamma(k+1) \tag{4.26a}$$

$$\textbf{subject to} \tag{4.26b}$$

$$\triangle\mathcal{Q}_1 : \begin{cases} \gamma(k+1) \geq 0 \\ \triangle L(\mathbf{x}(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma(k+1)I_N \end{cases} \tag{4.26c}$$

$$\triangle\mathcal{Q}_2 : \begin{cases} \mathcal{Q}_{2.1}: & \triangle f_d(x_i(k+1), x_j(k+1)) > \rho_1, \\ & \qquad \forall (i,j) \in \mathcal{E} \\ \mathcal{Q}_{2.2}: & \mathbf{x}_i(k+1) \in \mathcal{F}_i, \quad i = 1,\dots,N \\ \mathcal{Q}_{2.3}: & \mathbf{u}_i(k) \in \mathcal{U}_i, \quad i = 1,\dots,N \\ \mathcal{Q}_{2.4}: & \mathbf{x}_i(k+1) = \mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k)),\ i = 1,\dots,N \end{cases} \tag{4.26d}$$

where, $\mathcal{S}_{\triangle\mathcal{Q}_2} = \{\rho_1, (A_{1i}, A_{2i}, b_{1i}, H_i, h_i)_{i=1,\dots,N}\}$. As a solution of (4.26) we find the optimal control inputs $\mathbf{u}_i(k) = (u_i(\tau)^\top, u_i(\tau+1)^\top)^\top$ that drive the system (4.16) from $\mathbf{x}_i(k)$ to $\mathbf{x}_i(k+1)$.

We define the concept of feasible state as follows.

Definition 4.1 *A state $\mathbf{x}(k)$ at time k is feasible if*

(i) $\mathbf{x}_i(k) \in \mathcal{F}_i$*, for all agents* i*,*

(ii) $\triangle L(x(k)) + \mathbf{1}_N\mathbf{1}_N^\top \succ 0$*,*

(iii) $d_{ij}^2(k) > \rho_1$*, for all* $(i,j) \in \mathcal{E}$*.*

For the optimization problem (4.26), as in (Kim and Mesbahi, 2006), we assume initial feasibility for the first time instance:

[2] In order to see this, consider the maximizing of $||A_{1i}v_i(\tau) - A_{1j}v_j(\tau)||$. This is equivalent to maximize the squared norm $||A_{1i}v_i(\tau) - A_{1j}v_j(\tau)||^2$, which is equivalent to the following non-convex quadratic program

$$\begin{array}{ll} \textbf{maximize} & \begin{pmatrix} v_i(\tau) \\ v_j(\tau) \end{pmatrix}^\top \begin{pmatrix} A_{1i}^\top A_{1i} & -A_{1i}^\top A_{1j} \\ -A_{1j}^\top A_{1i} & A_{1j}^\top A_{1j} \end{pmatrix} \begin{pmatrix} v_i(\tau) \\ v_j(\tau) \end{pmatrix} \\ \textbf{subject to} & v_i(\tau) \in \mathcal{F}_{v,i}, \quad v_j(\tau) \in \mathcal{F}_{v,j} \end{array}$$

Assumption 4.3 *The initial state* $\mathbf{x}(0)$ *is a feasible state.*

The following theorem states formally the persistent feasibility property:

Theorem 4.1 (Persistent Feasibility) *If for any discrete time* k*,* $\mathbf{x}(k)$ *is a feasible state according to Definition 4.1, then the problem* (4.26) *will be feasible for the discrete time* $k+1$.

Proof. Consider $\mathbf{x}_i(k+1) = (x_i(k)^\top, \mathbf{0}_3^\top)^\top$ as the solution of the optimization (4.26) at time $k+1$. This solution satisfies $\triangle\mathcal{Q}_1$, $\mathcal{Q}_{2.1}$, and $\mathcal{Q}_{2.2}$. Moreover, since $\mathbf{x}_i(k) \in \mathcal{F}_i$ by assumption, there exist control inputs $\mathbf{u}_i(k) \in \mathcal{U}_i$ for all the agents for which $(x_i(k)^\top, \mathbf{0}_3^\top)^\top = \mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k))$. Therefore the solution $\mathbf{x}_i(k+1)$ satisfies $\mathcal{Q}_{2.3}$ and $\mathcal{Q}_{2.4}$ and thus the claim. □

Combining Theorem 4.1 with Assumption 4.3, it follows that the sequence of problems (4.26) is feasible for all $k > 0$. We note that persistent feasibility (Theorem 4.1) is a fundamental property to guarantee that the overall optimization scheme remains feasible, while we show later (in the distributed case) that the sequence of solutions lead to a monotonic increase of the cost function.

We recall once again the reasons for the initial choices of $k = \tau/2$ and $\mathcal{F}_i$, which should appear clearer after Theorem 4.1. The fact that $\mathbf{x}_i(k) \in \mathcal{F}_i$ guarantees that the solution $\mathbf{x}_i(k+1) = (x_i(k)^\top, \mathbf{0}_3^\top)^\top$ is feasible in terms of admissible control action, which is a sufficient condition to guarantee that the optimization problem (4.26) is persistently feasible. The choice $k = \tau/2$ ensures that $\mathcal{F}_i$ is always non-empty.

The optimization procedure (4.26) described in this section finds a local optimum of the connectivity maximization problem in a centralized manner using linearization. In the next section, we describe an approach that allows the problem to be solved using local computation and limited communication resources, which increases the flexibility and practical applicability of the robotic network.

4.2.4 Distributed Solution for the Extended Problem

In the following we present a non-iterative distributed solution to solve (4.26). By non-iterative we mean here that we will use only one round of communication/computation among the different agents per optimization step k. We note that this is not a trivial task, since commonly used decomposition methods for optimization problems (if applicable, e.g. in (De Gennaro and Jadbabaie, 2006)) typically require iterative solutions (many rounds of communication/computation per optimization step k) which may not be amenable to fast real-time implementations.

Our solution depends on subproblems that each agent solves locally and whose size can be decided according to the available resources. This size is influenced by the notion of an enlarged neighborhood set, collecting all the agents whose data are available locally at each time step k. The proposed distributed solution is computed in two phases. The first step is to solve a local optimization problem in which the farthest agents (in terms of graph distance, i.e. minimum number of connecting edges) are constrained to be stationary, i.e. $\mathbf{x}_i(k+1) = (x_i(k)^\top, \mathbf{0}_3^\top)^\top$. This step is similar to a Jacobi-type optimization (Bertsekas and Tsitsiklis, 1997), where only certain variables are updated at a time. The second step is to share the proposed solutions within the enlarged neighborhood and combine them using an agent-dependent positive linear combination. We note that this sharing/combining procedure is performed just once for each optimization step, making the overall scheme non-iterative in contrast with commonly used consensus algorithms. The key point in the proposed distributed solution is to *jointly* construct the feasible local problems with modified local constraints *and* the positive linear combination of the solutions to preserve feasibility of the global solution and a monotonically increasing cost function.

Let $\mathcal{J}_i$ denote the enlarged neighborhood of i consisting of all the agents whose state is known by agent i at each sampling time k (either through direct or indirect communication[3]). We define this set in a recursive way: let $\mathcal{N}_i^1$ be the standard, first-order neighborhood of i, i.e. $\mathcal{N}_i^1 = \mathcal{N}_i^+$, then, the n_i-size enlarged neighborhood of i for $n_i > 1$ is defined as

$$\mathcal{J}_i = \mathcal{N}_i^{n_i} = \bigcup_{j \in \mathcal{N}_i^{n_i-1}} \mathcal{N}_j^{n_i-1}, \tag{4.27}$$

in other words, the collection of the $(n_i - 1)$-size enlarged neighborhoods of all $j \in \mathcal{N}_i^{n_i-1}$. The scalar $n_i \geq 1$ implies bounds on the diameter of the communication graph composed by the agents in $\mathcal{J}_i$. We will explain how the choice of n_i is made by the agents locally to trade-off computations/communications with respect to sub-optimality of the distributed solution.

The cardinality of $\mathcal{J}_i$ is J_i. We call the set of agents belonging to $\partial\mathcal{J}_i$, the bordering agents of $\mathcal{J}_i$ defined as

$$\partial\mathcal{J}_i = \{j | j \in \mathcal{J}_i, j \notin \mathcal{N}_i^{n_i-1}\} \tag{4.28}$$

Denote the graph Laplacian associated with the communication graph corresponding to the agents in $\mathcal{J}_i$ as L_{i,n_i} and the communication link set of $\mathcal{J}_i$ as $\mathcal{E}_{i,n_i}$. Figure 4.2 provides a graphical illustration of this notation for $n_i = 2$. Define $\mathbf{x}_{\mathcal{J}_i}$ and $\mathbf{u}_{\mathcal{J}_i}$ as the stacked

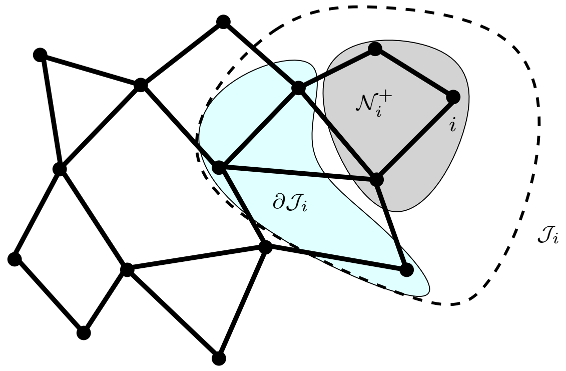


Figure 4.2: *Notation for the distributed solution in case the size of the enlarged neighborhood for agent i is $n_i = 2$. The thick lines represent links between connected agents.*

[3]Which also means that agents share all known states within their neighborhoods.

vectors collecting the states and the lifted control inputs for all the agents j belonging to the enlarged neighborhood of i, i.e. $j \in \mathcal{J}_i$.

As a first step of the distributed solution, for each agent i, we consider local modified problems $\triangle\mathbf{P}_i$ of the form:

$$\triangle\mathbf{P}_i(\triangle L_{i,n_i}(x_{\mathcal{J}_i}), \mathbf{x}_{\mathcal{J}_i}(k), \mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}) : \quad \underset{\mathbf{x}_{\mathcal{J}_i}(k+1), \mathbf{u}_{\mathcal{J}_i}(k), \gamma_i(k+1)}{\textbf{maximize}} \quad \gamma_i(k+1) \tag{4.29a}$$

$$\textbf{subject to} \tag{4.29b}$$

$$\triangle\mathcal{Q}_1 : \begin{cases} \gamma_i(k+1) \geq 0 \\ \triangle L_{i,n_i}(\mathbf{x}_{\mathcal{J}_i}(k+1)) + \mathbf{1}_{J_i}\mathbf{1}_{J_i}^T \succ \gamma_i(k+1) I_{J_i} \end{cases} \tag{4.29c}$$

$$\triangle\tilde{\mathcal{Q}}_{2i} : \begin{cases} \tilde{\mathcal{Q}}_{2.1} : & \triangle f_d(x_i(k+1), x_j(k+1)) > \tilde{\rho}_{1ij}, \\ & \qquad \forall (i,j) \in \mathcal{E}_{i,n_i} \\ \tilde{\mathcal{Q}}_{2.2} : & \mathbf{x}_j(k+1) \in \tilde{\mathcal{F}}_j = \mathcal{F}_j, \quad j \in \mathcal{J}_i \\ \tilde{\mathcal{Q}}_{2.3} : & \mathbf{u}_j(k) \in \tilde{\mathcal{U}}_j, \quad j \in \mathcal{J}_i \\ \tilde{\mathcal{Q}}_{2.4} : & \mathbf{x}_j(k+1) = \tilde{\mathcal{D}}_j(\mathbf{x}_j(k), \mathbf{u}_j(k)), \ j \in \mathcal{J}_i \end{cases} \tag{4.29d}$$

$$\mathcal{Q}_3 : \mathbf{x}_j(k+1) = (x_j(k)^\top, \mathbf{0}_3^\top)^\top, \qquad \text{for } j \in \partial\mathcal{J}_i \tag{4.29e}$$

where

- the dynamics $\tilde{\mathcal{D}}_j$ denotes a dynamical system of the same form as (4.18) but with the modified triplet $(\tilde{A}_{1j}, \tilde{A}_{2j}, \tilde{b}_{1j})$;
- the constraint $\tilde{\mathcal{U}}_j$ denotes a constraint of the same form as $\mathcal{U}_j$ but with the modified couple $(\tilde{H}_j, \tilde{h}_j)$;
- the set of the modified parameter is $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}} = \{(\tilde{\rho}_{1ij}, \tilde{A}_{1j}, \tilde{A}_{2j}, \tilde{b}_{1j}, \tilde{H}_j, \tilde{h}_j)_{j \in \mathcal{J}_i}\}$.

For now, the parameters in $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}$ could be thought of as arbitrary. However, we will show later (Theorem 4.2) how to construct the modified state matrices and parameter set $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}$ of constraint $\triangle\tilde{\mathcal{Q}}_{2i}$ to satisfy the constraints of the original linearized problem (4.26).

The optimal local decision variables (solution of $\triangle\mathbf{P}_i$) will be denoted as $\tilde{\gamma}_i(k+1)$, $\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1)$, and $\tilde{\mathbf{u}}_{\mathcal{J}_i}(k)$ respectively. We call $\tilde{\mathbf{x}}_{ij}(k+1)$ the state of agent j as computed by agent i and we use the same notation for $\tilde{\mathbf{u}}_{ij}(k)$. We note that the optimal local decision variables $\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1)$ and $\tilde{\mathbf{u}}_{\mathcal{J}_i}(k)$ are composed of $\tilde{\mathbf{x}}_{ij}(k+1)$ and $\tilde{\mathbf{u}}_{ij}(k)$ for each $j \in \mathcal{J}_i$. We emphasize that the extra constraint $\mathcal{Q}_3$ is an important requirement to guarantee feasibility, as will be explained in Theorem 4.5. We will also require $\tilde{\mathcal{F}}_i = \mathcal{F}_i$ for all the agents as a sufficient condition of persistent feasibility.

Consider the set of all agents p which include agent i in their local problems $\triangle\mathbf{P}_p$, i.e. $i \in \mathcal{J}_p$, and denote by $\mathcal{J}_i^* = \{p | i \in \mathcal{J}_p\}$. Since the enlarged neighborhood size n_i could differ from agent to agent (we remark that we will explain later how this will be selected locally by the agents), $\mathcal{J}_i^* \neq \mathcal{J}_i$.

As a second step of the distributed solution, we construct the position update based on the previous solution $\mathbf{x}(k)$ and a positive linear combination of the local position solutions

$\tilde{x}_{\mathcal{J}_i}(k)$ as:

$$x_i(k+1) = x_i(k) + \sum_{j \in \mathcal{J}_i^*} s_j \delta\tilde{x}_{ji}(k+1), \qquad i = 1, \dots, N \tag{4.30}$$

for any *user selected* arbitrary $s_j > 0$ and, where $\delta\tilde{x}_{ji}(k+1) = \tilde{x}_{ji}(k+1) - x_i(k)$. Define

$$\bar{s}_i = \sum_{j \in \mathcal{J}_i^*} s_j$$

and observe that $\bar{s}_i$ is in general not equal to one. Although it is more common to use a weighted average of local solutions in consensus-type problems, the linear combination (4.30) of our approach is crucial for feasibility as shown in Theorem 4.5. We note also that, although the scalar $\bar{s}_i$ could assume any strictly positive value, i.e., $\bar{s} > 0$, it is advisable to upper limit it as $\bar{s}_i \leq 1$ due to the linearization procedure. In fact, bigger $\bar{s}_i$ would question the validity of the Taylor expansions in the local problems. We will assume $0 < \bar{s}_i \leq 1$ in the rest of the section.

We prove the following lemma regarding the sum of local position solutions, which is instrumental for the subsequent theorems.

Lemma 4.2 *For arbitrary vectors $q_{ij} \in \mathbb{R}^3$ where (i,j) are neighbors (i.e., $(i,j) \in \mathcal{E}$), and for any $\delta\tilde{x}_{pi}(k+1), \delta\tilde{x}_{pj}(k+1)$ part of the optimal solutions of the local problems $\Delta\mathbf{P}_p$ in (4.29), with $p \in \mathcal{J}_i^*$ and $p \in \mathcal{J}_j^*$ respectively, the following equality holds:*

$$q_{ij}^\top \left(\sum_{p \in \mathcal{J}_i^*} s_p \delta\tilde{x}_{pi}(k+1) - \sum_{p \in \mathcal{J}_j^*} s_p \delta\tilde{x}_{pj}(k+1) \right) = \\ q_{ij}^\top \sum_{p \in \mathcal{J}_i^* \cap \mathcal{J}_j^*} \left(\delta\tilde{x}_{pi}(k+1) - \delta\tilde{x}_{pj}(k+1) \right) \tag{4.31}$$

Proof. The first term of the equality (4.31) can be divided into three parts: $p \in \mathcal{J}_i^* \cap \mathcal{J}_j^*$, $p \in \mathcal{J}_i^* \wedge p \notin \mathcal{J}_j^*$, and $p \in \mathcal{J}_j^* \wedge p \notin \mathcal{J}_i^*$. Since we are interested in the case when i and j are neighbors, we can make the key observations that:

$$p \in \mathcal{J}_i^* \wedge p \notin \mathcal{J}_j^* \Rightarrow i \in \partial\mathcal{J}_p \tag{4.32}$$

$$p \in \mathcal{J}_j^* \wedge p \notin \mathcal{J}_i^* \Rightarrow j \in \partial\mathcal{J}_p \tag{4.33}$$

Consider the first implication (4.32). If $p \in \mathcal{J}_i^*$, then i and p are separated by at most n_p links. Furthermore, if $p \notin \mathcal{J}_j^*$, then j and p are separated by at least $n_p + 1$ links. Since i and j are neighbors, it follows that the separation between i and p is exactly n_p links and therefore $i \in \partial\mathcal{J}_p$. The second implication (4.33) can be proven by similar arguments. The two implications (4.32)-(4.33) allow us to rewrite the first part of the equality (4.31) as:

$$q_{ij}^\top \sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} s_p(\delta\tilde{x}_{pi}(k+1)-\delta\tilde{x}_{pj}(k+1))+$$
$$\underbrace{q_{ij}^\top \sum_{p\in\mathcal{J}_i^*\wedge p\notin\mathcal{J}_j^*} s_p\delta\tilde{x}_{pi}(k+1)}_{=0} \underbrace{-q_{ij}^\top \sum_{p\in\mathcal{J}_j^*\wedge p\notin\mathcal{J}_i^*} s_p\delta\tilde{x}_{pj}(k+1)}_{=0}$$

where the last two terms are 0 due to (4.32)-(4.33) and the constraint $\mathcal{Q}_3$ of $\triangle\mathbf{P}_p$ in (4.29), which requires $\delta\tilde{x}_{pi}(k+1) = 0$ and $\delta\tilde{x}_{pj}(k+1) = 0$ for $i \in \partial\mathcal{J}_p$ and $j \in \partial\mathcal{J}_p$, respectively. $\square$

We are ready to construct the parameter set $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}$ which defines the local set of constraints $\triangle\tilde{\mathcal{Q}}_{2i}$.

Theorem 4.2 (Local constraints for global feasibility) *Taking for each i, the following choices:*

- *the local parameter set $\triangle\tilde{\mathcal{Q}}_{2i}$ in (4.29) as*

$$\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}} = \{(\tilde{\rho}_{1ij}, \bar{s}_j^{-1}A_{1j}, A_{2j}, \bar{s}_j b_{1j}, H_j, \bar{s}_j^{-1}h_j)_{j\in\mathcal{J}_i}\}$$

  meaning,

$$\tilde{A}_{1j} = \bar{s}_j^{-1}A_{1j}, \quad \tilde{A}_{2j} = A_{2j}, \quad \tilde{b}_{1j} = \bar{s}_j b_{1j}, \quad \tilde{H}_j = H_j, \quad \tilde{h}_j = \bar{s}_j^{-1}h_j$$

  and

$$\tilde{\rho}_{1ij} = \bar{s}_{ij}^{-1}\left(\rho_1 + d_{ij}^2(k)\left(\bar{s}_{ij}-1\right)\right) \tag{4.34}$$

  with $\bar{s}_{ij} = \sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} s_p$;

- *the positive linear combination of the local optimal control inputs $\tilde{\mathbf{u}}_{ji}(k)$ in (4.29) as*

$$\mathbf{u}_i(k) = \sum_{j\in\mathcal{J}_i^*} s_j\tilde{\mathbf{u}}_{ji}(k) \tag{4.35}$$

- *the positive linear combination of the local optimal velocities $\tilde{v}_{ji}(k+1)$ in (4.29) as*

$$v_i(k+1) = \frac{\sum_{j\in\mathcal{J}_i^*} s_j\tilde{v}_{ji}(k+1)}{\bar{s}_i} \tag{4.36}$$

ensure that the updated position vector $x(k+1)$, the control vector $\mathbf{u}(k)$, and velocity vector $v(k+1)$ based on (4.30), (4.35), and (4.36) respectively, satisfy the set of constraints $\triangle\mathcal{Q}_2$ of the global problem (4.26).

Proof. The local constraints for the subproblem $\triangle\mathbf{P}_p$ in (4.29) are the following:

$$\text{for all } p \in \mathcal{J}_i^* \cap \mathcal{J}_j^* :$$
$$\tilde{\mathcal{Q}}_{2.1} : \quad \triangle f_d(x_i(k+1), x_j(k+1)) = d_{ij}^2(k) + {c_{ij}^d}^\top (\delta\tilde{x}_{pi}(k+1) - \delta\tilde{x}_{pj}(k+1)) > \tilde{\rho}_{1ij}, \tag{4.37a}$$

$$\text{for all } p \in \mathcal{J}_i^* :$$
$$\tilde{\mathcal{Q}}_{2.2} : \quad \tilde{\mathbf{x}}_{pi}(k+1) \in \tilde{\mathcal{F}}_i = \mathcal{F}_i \tag{4.37b}$$
$$\tilde{\mathcal{Q}}_{2.3} : \quad \tilde{\mathbf{u}}_{pi}(k) \in \tilde{\mathcal{U}}_i \tag{4.37c}$$
$$\tilde{\mathcal{Q}}_{2.4} : \quad \tilde{\mathbf{x}}_{pi}(k+1) = \tilde{\mathcal{D}}_i(\mathbf{x}_i(k), \tilde{\mathbf{u}}_{pi}(k)) \tag{4.37d}$$

The theorem claims that using the specified choice for $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}$, if we combine the local optimal solutions $(\tilde{\mathbf{x}}_{pi}(k+1), \tilde{\mathbf{u}}_{pi}(k))$ which satisfy the local constraints (4.37), using the positive linear combinations (4.30), (4.35), and (4.36) we will obtain a couple $(\mathbf{x}(k+1), \mathbf{u}(k))$ that satisfies the constraint $\triangle\mathcal{Q}_2$ of the global problem (4.26). This is what we need to prove.

Consider $\tilde{\mathcal{Q}}_{2.1}$ in (4.37a) and the positive linear combination for $x(k+1)$ in (4.30). By Lemma 4.2

$$\begin{aligned} d_{ij}^2(k) + {c_{ij}^d}^\top (\delta x_i(k+1) - \delta x_j(k+1)) &= \\ = d_{ij}^2(k) + \sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} {c_{ij}^d}^\top s_p(\delta\tilde{x}_{pi}(k+1) - \delta\tilde{x}_{pj}(k+1)) &> \\ (1-\bar{s}_{ij})d_{ij}^2(k) + \bar{s}_{ij}\tilde{\rho}_{1ij} \end{aligned} \tag{4.38}$$

The global position vector $x(k+1)$ is required to satisfy the global constraint:

$$d_{ij}^2(k) + {c_{ij}^d}^\top (\delta x_i(k+1) - \delta x_j(k+1)) > \rho_1 \tag{4.39}$$

which can be accomplished by selecting $\tilde{\rho}_{1ij}$ such that:

$$(1-\bar{s}_{ij})d_{ij}^2(k) + \bar{s}_{ij}\tilde{\rho}_{1ij} = \rho_1 \tag{4.40}$$

This gives the formula for $\tilde{\rho}_{1ij}$ in (4.34).

Consider the constraints $\tilde{\mathcal{Q}}_{2.4}$ in (4.37d) on the agents' dynamics. For the positive linear combination (4.30) the combined system dynamics becomes

$$\begin{pmatrix} x_i(k+1) \\ \sum_{p\in\mathcal{J}_i^*} s_p\tilde{v}_{pi}(k+1) \end{pmatrix} = \begin{pmatrix} I_3 & \bar{s}_i\tilde{A}_{1i}(I_3+\tilde{A}_{2i}) \\ 0_3 & \bar{s}_i\tilde{A}_{2i}^2 \end{pmatrix} \begin{pmatrix} x_i(k) \\ v_i(k) \end{pmatrix} + \begin{pmatrix} \tilde{b}_{1i}\tilde{A}_{1i} & 0_3 \\ \tilde{b}_{1i}\tilde{A}_{2i} & \tilde{b}_{1i}I_3 \end{pmatrix} \sum_{p\in\mathcal{J}_p^*} s_p\tilde{\mathbf{u}}_{pi}(k) \tag{4.41}$$

Since the agents have to move according to the dynamical system (4.18) encoded in the global constraint $\mathcal{Q}_{2.4}$ of (4.26), the update (4.41) and the state equation (4.18) have to be the same. It is not difficult to see that this is ensured by the choice $\tilde{A}_{1i} = \bar{s}_i^{-1} A_{1i}$, $\tilde{A}_{2i} = A_{2i}$, $\tilde{b}_{1i} = \bar{s}_i b_{1i}$, and the linear combinations (4.35) and (4.36) for the local control inputs $\tilde{\mathbf{u}}_{pi}(k)$ and local velocities $\tilde{v}_{pi}(k+1)$.

From the linear combination on the control (4.35) and the global constraint $\mathcal{Q}_{2.3}$ in (4.26) follows the specification for the local constraint $\tilde{\mathcal{Q}}_{2.3}$ in (4.29):

$$\tilde{\mathcal{U}}_i = \{\tilde{\mathbf{u}}_{pi}(k) \in \mathbb{R}^3 | H_i \tilde{\mathbf{u}}_{pi} \leq \bar{s}_i^{-1} h_i\} \tag{4.42}$$

from which $(\tilde{H}_i, \tilde{h}_i) = (H_i, \bar{s}_i^{-1} h_i)$. We recall that the positive linear combination on the control input (4.35) has been constructed in a way to steer the system (4.18) from the position $x(k)$ to the updated position $x(k+1)$ in (4.30) while respecting the global constraints $\mathcal{Q}_{2.3}$ in (4.26).

Consider now $\tilde{\mathcal{Q}}_{2.2}$ in (4.29). We need to prove that if the local optimal states $\tilde{\mathbf{x}}_{pi}(k+1)$ belong to the set $\tilde{\mathcal{F}}_i$ in (4.29), then the updated state $\mathbf{x}_i(k+1)$ constructed via the linear combinations on position (4.30) and velocity (4.36) belongs to the set $\mathcal{F}_i$ as expressed in the global constraint $\mathcal{Q}_{2.2}$ in (4.26). First of all, it is straightforward to see that the local inequalities

$$-\begin{pmatrix} \tilde{H}_i \tilde{b}_{1i}^{-1}(I_3 + \tilde{A}_{2i}) \\ \tilde{H}_i \tilde{b}_{1i}^{-1} \tilde{A}_{2i}(I_3 + 2\tilde{A}_{2i}) \end{pmatrix} \tilde{v}_{pi}(k+1) \leq \begin{pmatrix} \tilde{h}_i \\ \tilde{h}_i \end{pmatrix} \tag{4.43}$$

are equivalent to the inequalities (4.22), meaning that by construction $\tilde{\mathcal{F}}_i = \mathcal{F}_i$. Recall that the set $\mathcal{F}_i$ does not constrain the position. Since the updated velocity $v_i(k+1)$ in (4.36) is obtained by a positive linear combination of local $\tilde{v}_{pi}(k+1)$ then also $v_i(k+1)$ will satisfy the inequalities (4.43), and therefore the updated state $\mathbf{x}_i(k+1)$ belongs to $\mathcal{F}_i$.

Having ensured that with the choices of Theorem 4.2 the positive linear combinations of the local solutions satisfy the constraints $\mathcal{Q}_{2.1} - \mathcal{Q}_{2.4}$ of (4.26), Theorem 4.2 is proven. $\square$

Theorem 4.2 not only gives a procedure to construct the local constraints so that the linear combination (4.30) satisfies the global constraints, it also establishes a link between the local quantities and the global ones. Furthermore, it ensures that in order to move to the updated state $\mathbf{x}_i(k+1)$ each agent can implement the linear combination of the lifted control input (4.35) as summarized in Algorithm 4.1.

4.2.5 Properties of the Distributed Solution

In the previous section we have seen how to construct the local problem parameter set $\mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}$ and positive linear combinations of the local solutions to ensure that the combined solution $(\mathbf{x}(k+1), \mathbf{u}(k))$ satisfies the constraint $\triangle\mathcal{Q}_2$ of the global problem (4.15a). In this section we will look at the connectivity constraint $\triangle\mathcal{Q}_1$ and at the persistent feasibility of Algorithm 4.1. In particular we claim that

(C1) The algebraic connectivity of the global linearized Laplacian $\triangle L(x(k+1))$ of (4.15a)

Algorithm 4.1 Distributed λ_2 Maximization

```
1: Input for each agent $i$: $\mathbf{x}_j(k),\ j \in \mathcal{J}_i$
   ▷ Available data:   $f_w, f_d, \mathcal{J}_i, (\mathcal{U}_j|\text{for } j \in \mathcal{J}_i), (\mathcal{D}_j|\text{for } j \in \mathcal{J}_i), \rho_1, (s_i|\text{for } i \in N)$
2: Solve $\triangle\mathbf{P}_i$ in (4.29) computing $(\tilde{\mathbf{x}}_{ji}(k+1), \tilde{\mathbf{u}}_{ji}(k+1)),\ j \in \mathcal{J}_i$
   $\triangle\mathbf{P}_i(\triangle L_{i,n_i}(x_{\mathcal{J}_i}), \mathbf{x}_{\mathcal{J}_i}(k), \mathcal{S}_{\triangle\tilde{\mathcal{Q}}_{2i}}) : \underset{\mathbf{x}_{\mathcal{J}_i}(k+1), \mathbf{u}_{\mathcal{J}_i}(k), \gamma_i(k+1)}{\textbf{maximize}} \quad \gamma_i(k+1)$
   $\textbf{subject to}$
   $\triangle\mathcal{Q}_1 : \begin{cases} \gamma_i(k+1) \geq 0 \\ \triangle L_{i,n_i}(\mathbf{x}_{\mathcal{J}_i}(k+1)) + \mathbf{1}_{J_i}\mathbf{1}_{J_i}^T \succ \gamma_i(k+1) I_{J_i} \end{cases}$
   $\triangle\tilde{\mathcal{Q}}_{2i} : \begin{cases} \tilde{\mathcal{Q}}_{2.1} : & \triangle f_d(x_i(k+1), x_j(k+1)) > \tilde{\rho}_{1ij}, \\ & \quad \forall (i,j) \in \mathcal{E}_{i,n_i} \\ \tilde{\mathcal{Q}}_{2.2} : & \mathbf{x}_j(k+1) \in \tilde{\mathcal{F}}_j = \mathcal{F}_j, \quad j \in \mathcal{J}_i \\ \tilde{\mathcal{Q}}_{2.3} : & \mathbf{u}_j(k) \in \tilde{\mathcal{U}}_j, \quad j \in \mathcal{J}_i \\ \tilde{\mathcal{Q}}_{2.4} : & \mathbf{x}_j(k+1) = \tilde{\mathcal{D}}_j(\mathbf{x}_j(k), \mathbf{u}_j(k)),\ j \in \mathcal{J}_i \end{cases}$
   $\mathcal{Q}_3 : \mathbf{x}_j(k+1) = (x_j(k)^\top, \mathbf{0}_3^\top)^\top, \qquad \text{for } j \in \partial\mathcal{J}_i$
3: Communicate $\tilde{\mathbf{u}}_{ji}(k+1)$ among members of $\mathcal{J}_i$
4: Compute the positive linear combination:
   $\mathbf{u}_i(k) = \sum_{j \in \mathcal{J}_i^*} s_j \tilde{\mathbf{u}}_{ji}(k)$
5: Implement the control action $\mathbf{u}_i(k)$
```

with $x(k+1)$ computed via (4.30) is monotonically increasing[4] in each iteration, which implies that $x(k+1)$ will also satisfy $\triangle\mathcal{Q}_1$ of the global problem (4.15a) for a certain value of $\gamma(k+1) \geq \gamma(k)$.

(C2) The distributed optimization problem in Algorithm 4.1 is persistently feasible using the constructed $\triangle\tilde{\mathcal{Q}}_{2i}$'s in Theorem 4.2.

We will prove these claims in two steps: Theorem 4.3 and 4.4 establish (C1), by linking the linear combination (4.30) and the algebraic connectivity through the linear dependence of the linearized Laplacian on the position x. The constraint $\mathcal{Q}_3$ plays a crucial role here to ensure the feasibility of the local solutions. Theorem 4.5 shows that property (C2) holds, by the use of the relation between local and global feasibility of Theorem 4.2.

First of all consider the linearized Laplacian $\triangle L(x(k+1))$, we recall that its entry (i,j) has the expression

$$[\triangle L(x(k+1))]_{ij} = -w_{ij}(k+1) = -w_{ij}(k) - c_{ij}^{w\top}(\delta x_i(k+1) - \delta x_j(k+1)).$$

For this reason we can rewrite $\triangle L(x(k+1))$ as a sum

$$\triangle L(x(k+1)) = \triangle L(\delta x(k+1)) + L(x(k)).$$

[4]By the term monotonically increasing we mean that $\lambda_2(x(k+1)) \geq \lambda_2(x(k))$, while we indicate with strictly monotonically increasing the relation $\lambda_2(x(k+1)) > \lambda_2(x(k))$. Note that in some references alternative definitions can be found, for example the relation $\lambda_2(x(k+1)) \geq \lambda_2(x(k))$ can be called monotonically non-decreasing, while $\lambda_2(x(k+1)) > \lambda_2(x(k))$ as monotonically increasing.

Under the validity of the employed Taylor approximation, we assume that for all practical situations the value of $L(x(k))$ is equivalent to its linearized approximation $\triangle L(x(k))$, and therefore we can write

$$\triangle L(x(k+1)) = \triangle L(\delta x(k+1)) + \triangle L(x(k)) = L(x(k+1)) \tag{4.44}$$

We will further comment on this approximation in Section 4.5.

Consider the local problem $\triangle \mathbf{P}_i$ in (4.29), and its solution comprised of $\tilde{x}_{ij}(k+1)$ for all $j \in \mathcal{J}_i$. Construct the global vector $\tilde{x}^{(i)}(k+1)$ whose entries are determined based on the local solution as

$$\tilde{x}^{(i)}(k+1) = \begin{bmatrix} \tilde{x}_1^{(i)}(k+1) \\ \vdots \\ \tilde{x}_N^{(i)}(k+1) \end{bmatrix},$$

$$\text{with } \tilde{x}_j^{(i)}(k+1) = \begin{cases} \tilde{x}_{ij}(k+1) & \text{if } j \in \mathcal{J}_i \\ x_j(k) & \text{otherwise} \end{cases} \tag{4.45}$$

where we keep those agent positions that have not been optimized fixed, and we update the rest from the solution of the local problem, as in a Jacobi-type optimization approach (Bertsekas and Tsitsiklis, 1997). We can prove the following theorem.

Theorem 4.3 (C1.a) *The positions $\tilde{x}^{(i)}(k+1)$ in (4.45) constructed from the solution of the local problem $\triangle \mathbf{P}_i$ in (4.29), monotonically increase the algebraic connectivity of the Laplacian matrix:*

$$\triangle L(\tilde{x}^{(i)}(k+1)) \succeq \triangle L(x(k)). \tag{4.46}$$

Proof. Since $\triangle L$ depends linearly on the position x by (4.44) we can write

$$\triangle L(\tilde{x}^{(i)}(k+1)) = \triangle L(\delta\tilde{x}^{(i)}(k+1)) + \triangle L(x(k)),$$

thus the relation (4.46) can be interpreted as

$$\triangle L(\delta\tilde{x}^{(i)}(k+1)) \succeq 0. \tag{4.47}$$

We recall that,

First: for (4.45) $\delta\tilde{x}_j^{(i)}(k+1) = 0$ if $j \notin \mathcal{J}_i$.

Second: for the constraint $\mathcal{Q}_3$ in the local problem $\triangle \mathbf{P}_i$ (4.29), $\delta\tilde{x}_j^{(i)}(k+1) = 0$ if $j \in \partial\mathcal{J}_i$.

For these two observations, $[\triangle L(\delta\tilde{x}^{(i)}(k+1))]_{ij} \neq 0$ only if $(i,j) \in \mathcal{E}_{i,n_i}$ and therefore up to a reodering the Laplacian $\triangle L(\delta\tilde{x}^{(i)}(k+1))$ has the form

$$\left[\begin{array}{c|c} \triangle L_{i,n_i}(\delta\tilde{x}_{\mathcal{J}_i}(k+1)) & 0 \\ \hline 0 & 0 \end{array}\right] \succeq 0. \tag{4.48}$$

We recall that $\tilde{x}_{\mathcal{J}_i}(k+1)$ is the optimal decision variable for the position in the local optimization problems (and the order of the single elements is not important).

We can now restate (4.47) via (4.48) as

$$\triangle L_{i,n_i}(\delta\tilde{x}_{\mathcal{J}_i}(k+1)) \succeq 0$$

or

$$\triangle L_{i,n_i}(\tilde{x}_{\mathcal{J}_i}(k+1)) \succeq \triangle L_{i,n_i}(\tilde{x}_{\mathcal{J}_i}(k))$$

which is true due to the local optimality of the local solution of $\triangle\mathbf{P}_i$. □

Furthermore, we can relate the positions $\tilde{x}^{(i)}(k+1)$ in (4.45) with $x_i(k+1)$ in (4.30), which is crucial for the proof of the monotonically increasing property (C1).

Lemma 4.3 *When considering the positions $\tilde{x}^{(i)}(k+1)$ in (4.45) and $x_i(k+1)$ in (4.30) the following equality holds:*

$$\triangle L(\delta x(k+1)) = \sum_{i=1}^{N} s_i \triangle L(\delta\tilde{x}^{(i)}(k+1)) \tag{4.49}$$

Proof. Let us consider the entry (i,j) of the Laplacian $\triangle L$ on both sides of the expression (4.49) (indicated as ℓ_{ij}). For the right side, ℓ_{ij}^{right} can be expressed as

$$\ell_{ij}^{\text{right}} = c_{ij}^{w\top} \sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} s_p(\delta\tilde{x}_{pi}(k+1) - \delta\tilde{x}_{pj}(k+1))$$

since the entry (i,j) will exist only for the subproblems $\triangle\mathbf{P}_p$ with $p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*$. For the left side,

$$\ell_{ij}^{\text{left}} = c_{ij}^{w\top}(\delta x_i(k+1) - \delta x_j(k+1)) = \\ c_{ij}^{w\top}\left(\sum_{p\in\mathcal{J}_i^*} s_p\delta\tilde{x}_{pi}(k+1) - \sum_{p\in\mathcal{J}_j^*} s_p\delta\tilde{x}_{pj}(k+1)\right)$$

The coefficient $c_{ij}^{w\top}$ is non-zero only if (i,j) are neighbors and using Lemma 4.2 leads to

$$\ell_{ij}^{\text{left}} = c_{ij}^{w\top} \sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} s_p(\delta\tilde{x}_{pi}(k+1) - \delta\tilde{x}_{pj}(k+1))$$

□

Using Theorem 4.3 and Lemma 4.3 we can now prove the monotonically increasing property of the algebraic connectivity of the global linearized Laplacian $\triangle L(x(k+1))$, formally stated in Theorem 4.4.

Theorem 4.4 (C1.b) *The algebraic connectivity of the global linearized Laplacian $\triangle L(x(k+1))$ is monotonically increasing in each iteration, meaning $\triangle L(x(k+1)) \succeq \triangle L(x(k))$, where $x(k+1)$ is computed by the combination (4.30).*

Proof. Theorem 4.3 implies $\triangle L(\delta\tilde{x}^{(i)}(k+1)) \succeq 0$ for all i. Thus summing over all agents leads to

$$\sum_{i=1}^{N} s_i \triangle L(\delta\tilde{x}^{(i)}(k+1)) \succeq 0$$

Considering the linear combination $x_i(k+1)$ in (4.30), and the associated global vector $x(k+1)$, by Lemma 4.3 it follows that $\triangle L(\delta x(k+1)) \succeq 0$. From the linear dependence of $\triangle L$ on x (Equation (4.44)),

$$\triangle L(x(k+1)) = \triangle L(\delta x(k+1)) + \triangle L(x(k))$$

and therefore it follows that $\triangle L(x(k+1)) - \triangle L(x(k)) \succeq 0$ and the desired property: $\triangle L(x(k+1)) \succeq \triangle L(x(k))$. □

Finally, we can show the persistent feasibility of the distributed optimization algorithm presented in Algorithm 4.1.

Theorem 4.5 (C2) *The distributed optimization algorithm presented in Algorithm 4.1 is persistently feasible.*

Proof. We have to prove that if, for any discrete time k, $\mathbf{x}(k)$ is a feasible initial state for the global optimization problem $\triangle\mathbf{P}$ (4.26) at the discrete time k (Definition 4.1), then there will be a feasible solution to the distributed optimization problem in Algorithm 4.1. Such a feasible solution can be thought of as an initial state $\mathbf{x}(k+1)$ for the global optimization problem $\triangle\mathbf{P}$ (4.26) at the discrete time $k+1$. We prove the existence of such feasible solution in two steps.

Step 1. Using the assumption that $\mathbf{x}(k)$ is a feasible initial state for the global optimization problem $\triangle\mathbf{P}$ (4.26) at time step k, we can show that $\mathbf{x}(k)$ is also a feasible initial state for the local problems $\triangle\mathbf{P}_i$ (4.29), which therefore are feasible and deliver local solutions $(\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1), \tilde{\mathbf{u}}_{\mathcal{J}_i}(k+1))$ satisfying the constraints $\triangle\mathcal{Q}_1$, $\triangle\tilde{\mathcal{Q}}_{2i}$, and $\mathcal{Q}_3$. This claim follows from Theorem 4.2, in particular from the fact that $\tilde{\rho}_{1ij} \le \rho_1$. In fact, from the assumption $\bar{s} \le 1$ and $d_{ij}^2(k) > \rho_1$ (feasibility at k), the relation (4.34) yields $\tilde{\rho}_{1ij} \le \rho_1$, and thus $\mathbf{x}(k)$ is also a feasible initial state for the local problems $\triangle\mathbf{P}_i$ (4.29).

Step 2. We can show that after merging/combining the resulting local solutions $(\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1), \tilde{\mathbf{u}}_{\mathcal{J}_i}(k+1))$, the final distributed state solution $\mathbf{x}(k+1)$ will be a feasible initial state for the global optimization problem $\triangle\mathbf{P}$ in (4.26) at the discrete time $k+1$. This second step follows directly from Theorem 4.2 and Theorem 4.4. □

Similarly to Theorems 4.2 and 4.4, we note that Theorem 4.5 holds even if the agents change the size of their enlarged neighborhood n_i from time step k to $k+1$, since the feasibility of the state in the local problems does not depend on the enlarged neighborhood size of $\mathcal{J}_i$. This fact will be used in the next section to allow adjusting the communication load of each agent and make Algorithm 4.1 adaptive.

4.2.6 Adapting the Communication Load

In this section we investigate further the properties of the distributed solution presented in Section 4.2.4. First we show in Theorem 4.6 that if all-to-all communication is allowed then the distributed solution of Algorithm 4.1 is equivalent[5] to the centralized approach in (4.26). Then we prove in Theorem 4.7 that starting from the same state vector $\mathbf{x}(k)$, if we run Algorithm 4.1 with different enlarged neighborhood sizes, the solution that delivers a higher algebraic connectivity at time step $k+1$ is the one with a larger neighborhood size n. This last fact enables us to characterize a local relative sub-optimality measure with respect to an enlarged neighborhood size.

Theorem 4.6 (Equivalence) *The distributed solution of Algorithm 4.1 is equivalent to the centralized one of* (4.26)*, if all-to-all communication is allowed (meaning $n_i = N$, $\forall i$, and thus there are no bordering agents) and if $s_i = 1/N$, $\forall i$, is chosen as weight in the positive linear combinations of the local states and inputs* (4.30)*,* (4.36)*, and* (4.35)*.*

Proof. Consider $n_i = N$, $\partial\mathcal{J}_i = \emptyset$ for all the agents, and the choice $s_i = 1/N$, $\forall i$. We have $\bar{s}_i = 1$ and $\sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_i^*} s_p = 1$. Therefore, as a consequence of the choices of Theorem 4.2, $\triangle\tilde{\mathcal{Q}}_{2i} \equiv \triangle\mathcal{Q}_2$. Furthermore, all the constructed local solutions $\tilde{x}^{(i)}(k+1)$ in (4.45) are the same and they are equivalent to the solution of the centralized problem $x(k+1)$ in (4.26). Given the specified selection of s_i, also the linear combination (4.30) is equivalent to $\tilde{x}^{(i)}(k)$ and therefore the distributed position solution delivered by Algorithm 4.1 is equivalent to the centralized one of (4.26). Since the same arguments hold for the control inputs and velocities the claim is proven. $\square$

We define the global position vector obtained using an enlarged neighborhood size n_i in the local problem $\triangle\mathbf{P}_i$ (4.29) as follows.

Definition 4.2 *The vector $x^{(i)}(k+1)\big|_{n_i}$ is the global position vector constructed from the local solutions as in* (4.45) *using an enlarged neighborhood size n_i in the local problem $\triangle\mathbf{P}_i$* (4.29)*.*

Furthermore, consider each of the local solutions $\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1)$, which are computed using different enlarged neighborhood sizes $n_1,\dots,n_N$ in the local problems $\triangle\mathbf{P}_i$ (4.29), with $i=1,\dots,N$. Let $\mathbf{n} = (n_1,\dots,n_N)$. The global solution using the local $\tilde{\mathbf{x}}_{\mathcal{J}_i}(k+1)$ can be redefined as follows, highlighting the dependence on the choice of $\mathbf{n}$.

Definition 4.3 *The vector $x(k+1)|_{\mathbf{n}}$, with $\mathbf{n} \in \mathbb{N}^N$, is the global solution of Algorithm 4.1 at step $k+1$, for the choice $\mathbf{n} = (n_1,\dots,n_N)$.*

Using the above definitions, we can prove the following theorem about the effect of an increased neighborhood size on the resulting algebraic connectivity.

Theorem 4.7 *If $\mathbf{n}_1 \geq \mathbf{n}_2$ element-wise, then the algebraic connectivity of $\triangle L\left(x(k+1)|_{\mathbf{n}_2}\right)$ is greater than or equal to the one of $\triangle L\left(x(k+1)|_{\mathbf{n}_1}\right)$, implying $\triangle L\left(x(k+1)|_{\mathbf{n}_2}\right) \succeq \triangle L\left(x(k+1)|_{\mathbf{n}_1}\right)$.*

[5] Meaning that the two solutions (centralized and distributed) are the same.

Proof. By optimality and due to the linearity of L on x (Eq. (4.44)), for each i we can state

$$\triangle L\left(\delta x^{(i)}(k+1)\Big|_{\mathbf{n}_{2,i}}\right) \succeq \triangle L\left(\delta x^{(i)}(k+1)\Big|_{\mathbf{n}_{1,i}}\right)$$

Multiplying by s_i and summing over i leads to

$$\sum_{i=1}^{N} s_i \triangle L\left(\delta x^{(i)}(k+1)\Big|_{\mathbf{n}_{2,i}}\right) \succeq \sum_{i=1}^{N} s_i \triangle L\left(\delta x^{(i)}(k+1)\Big|_{\mathbf{n}_{1,i}}\right)$$

By Lemma 4.3 the claim follows. □

We note that Theorem 4.7 holds from k to $k+1$. Due to the non-linear/non-convex nature of the original problem (4.8a), this result does not hold in general from k to $k+2$ or beyond, as we will show in the simulation experiments of Section 4.2.7.

Theorem 4.7 is instrumental to construct a measure that can be used to decide locally on-line whether to increase or decrease the size n_i of the enlarged neighborhood. This measure can be used to adapt n_i to influence the trade-off between the increase of the algebraic connectivity or the reduction of the communication cost. For this purpose, we define two local relative sub-optimality measures with respect to an enlarged neighborhood of larger size as

$$e_i^{+} = 1 - \frac{\lambda_2(\triangle L_{i,n_i+1}(x^{(i)}(k+1)\big|_{n_i}))}{\lambda_2(\triangle L_{i,n_i+1}(x^{(i)}(k+1)\big|_{n_i+1}))}$$

$$e_i^{-} = 1 - \frac{\lambda_2(\triangle L_{i,n_i}(x^{(i)}(k+1)\big|_{n_i-1}))}{\lambda_2(\triangle L_{i,n_i}(x^{(i)}(k+1)\big|_{n_i}))}$$

which determine the sub-optimality of the local solutions (4.45) with n_i+1 and n_i-1 with respect to the one obtained with n_i. In particular, e_i^{+} measures the gain, in terms of local algebraic connectivity, one would have by increasing the enlarged neighborhood size from n_i to n_i+1, while e_i^{-} measures the loss of local algebraic connectivity going from n_i to n_i-1. (We note that both e_i^{+} and e_i^{-} are non-negative due to Theorem 4.7).

Given specific lower/upper thresholds for e_i^{+} and e_i^{-} the agents can decide locally to increase or decrease n_i at the successive time step k, trading off increased communication efforts (for larger n_i) to smaller local algebraic connectivity increases (for smaller n_i), making Algorithm 4.1 adaptive. We note that although these sub-optimality measures are local, changing n_i locally by each agent has an effect on the global solution as illustrated by the relation (4.63) in Lemma 4.3. We note also that in order to compute e_i^{+} and e_i^{-} it is necessary to solve three optimization problems of the kind (4.29) for each i. Since this can be computationally expensive, the agents can decide to determine e_i^{+} and e_i^{-} only once in a given number of discrete time steps.

Remark 4.4 *We remark that Theorem 4.7 as well as the other lemmas and theorems are valid under the original assumption that the agents are perfectly synchronized. Future research directions encompass the possible asynchronism in the agents' clocks.*

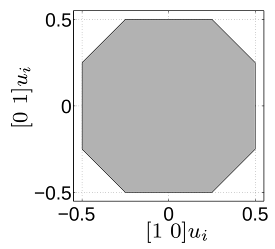


Figure 4.3: *Polytopic constraint for u_i. The shaded region represents the set $\bar{\mathcal{U}}_i \subset \mathbb{R}^2$.*

4.2.7 Simulation Results

In this section, we present numerical simulation results to illustrate how the proposed distributed algorithm performs with respect to the centralized scheme. We use a benchmark problem motivated by (Kim and Mesbahi, 2006). Our scenario considers $N = 10$ agents moving on a 2D plane initially placed close to the horizontal axis and forming a connected graph. The initial position vector is $x_i(0) = [-6.75 + 1.5(i-1),\ y_i]^\top$, where y_i is drawn from a Gaussian distribution, with mean 0 and standard deviation $\sigma = 0.1$. Randomness is added to test the algorithm's sensitivity to different initial conditions (due to the sequential convex programming approach). We consider the triples (A_{1i}, A_{2i}, b_{1i}) to be all equal to $(I_2\Delta t/2, 0.75 I_2, I_2\Delta t/2)$ with $\Delta t = 1$, while all the u_i's are constrained in the polytopic region of Figure 4.3.

The other simulation parameters include the weighting function of Figure 4.1, $\rho_1 = 0.75$, $\rho_2 = 3$ and final time $T_\mathrm{f} = 300$.

In Figures 4.4-4.5, an example of the trajectories using the centralized and the distributed solutions are depicted (all starting from the same initial configuration). In the adaptive case, we start with $n_i = 2$ for all agents and at every 5-th discrete time step k we compute the sub-optimality measures. If the gain in increasing the enlarged neighborhood size is high enough, i.e., $e_i^+ > 0.05$, we increase n_i, while if this gain is not high enough, i.e., $e_i^+ < 0.05$, and the losses in decreasing the neighborhood size are not too big, i.e., $e_i^- < 0.01$, we decrease n_i to reduce the communication and computation costs.

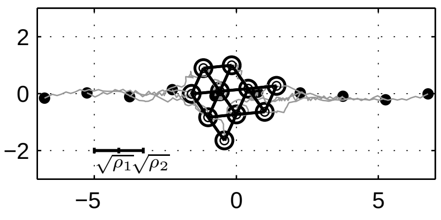


Figure 4.4: *Centralized solution: the initial positions are marked with black dots. The final positions are marked with circles. The bold lines represent the final communication graph and the thin lines the agent trajectories.*

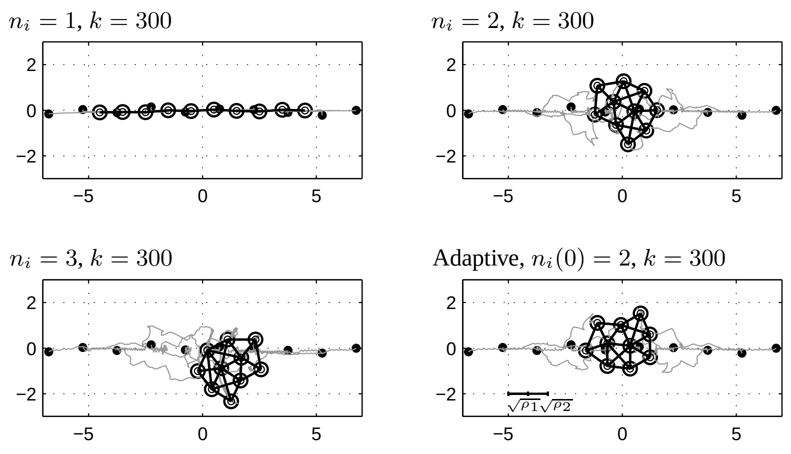


Figure 4.5: *Simulation results of the distributed approach for various local neighborhood sizes n_i (same for all the agents except in the adaptive case). The initial positions are marked with black dots. The final positions are marked with circles. The bold lines represent the final communication graph and the thin lines the agent trajectories. In the adaptive case, we start with $n_i = 2$ for all agents and at every 5-th discrete time step k we compute the sub-optimality measures. If $e_i^+ > 0.05$ we increase n_i, if $e_i^+ < 0.05$ and $e_i^- < 0.01$ we decrease n_i.*

Figure 4.6 shows, for the same simulation, the algebraic connectivity as a function of the sampling time k, and clearly illustrates the nonlinear/non-convex nature of the problem. In particular,

- Distributed and centralized solutions are based on different agents' trajectories and therefore their final achieved algebraic connectivities are not strictly related. It may happen, as in Figure 4.6, that the distributed approximation leads to a better final λ_2, or the contrary may happen (as for $n_i = 1$).
- The distributed solution converges slower than the centralized one to the final configuration. This was to be expected since the centralized solution has global knowledge about the robotic network. We recall that this final configuration is in general only a local maximum for the algebraic connectivity.

We perform 50 simulation runs varying the initial configuration of the agents. For each run, we compute the centralized and the distributed solutions and we compare their final connectivity, $\lambda_2^{\mathrm{centr}}$ and $\lambda_2^{\mathrm{distr}}$, respectively. We report the results in Table 4.1. For better comparison, we report that in the adaptive case $n_i = 2.2$ on average, with a maximum of $n_i = 5$. We can observe that

- Different choices of the local neighborhood sizes n_i affect the final achieved λ_2. In particular, for the choice $n_i = 1$, the agents perform significantly worse than for other n_i.

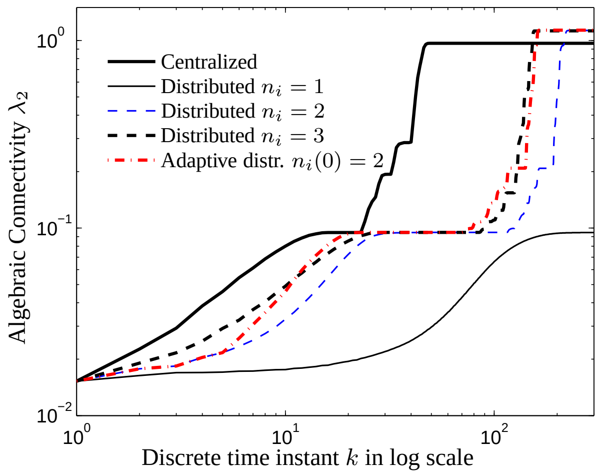


Figure 4.6: *Algebraic connectivity as a function of time k for both the centralized and the distributed solutions of Figure 4.4 and 4.5.*

- Using the adaptive case, the final λ_2 is comparable with the centralized solution in most of the simulations (or even better, due to the nonlinear nature of the problem). This is an important point, since the adaptive case use an enlarged neighborhood size of $n_i = 2.2$ (on average) and still obtains performances close or better than the fixed choice $n_i = 3$.

To further assess the proposed distributed algorithm, we include in Table 4.1 simulation results for $N \in \{20, 40\}$ robots starting from a feasible random configuration (not necessarily on a line) and using the adaptive algorithm with $n_i(0) = 2$. Each of these cases has been run 50 times. We can observe that both in the $N = 20$ case (where the average n_i is 2.7) and in the $N = 40$ case (average $n_i = 2.6$), the results are in line with the conclusions we have drawn for the case of $N = 10$. In addition, an example of the trajectories and algebraic connectivity using the centralized and the distributed solutions is also depicted in Figure 4.7 and 4.8 to show the very similar final configuration. From these results one could conjecture both the scalability of Algorithm 4.1 (for the adaptive case) and its increased performances dealing with large systems.

In particular, while the number of agents passes from $N = 10$ to $N = 40$, the averaged size of the enlarged neighborhood stays rather the same (and also the performance in term of final λ_2). This means that the computational and communication efforts for the single agent stay the same (per step k). Thus, the gain of the distributed solution with respect to the centralized solution, in terms of computations and communications, increases.

Another important consideration could be that for an increased number of agents, the time to converge to the final configuration is higher than for a lower number of agents. This implies that the *total* communication/computation increases with the number of agents. However, from Figure 4.6 and 4.7 we expect that the ratio *total* communication/computation for the distributed case and the centralized one stays approximately the same when increasing the number of agents.

Ratio $\frac{\lambda_2^{\text{distr}}}{\lambda_2^{\text{centr}}}$	$N = 10$				$N = 20$	$N = 40$
	$n_i = 1$	$n_i = 2$	$n_i = 3$	$n_i(0) = 2$	$n_i(0) = 2$	$n_i(0) = 2$
$(0.1 - 0.3]$	50	26	0	0	0	0
$(0.3 - 0.8]$	0	0	5	4	3	3
$(0.8 - 1.0]$	0	12	22	21	21	24
$(1.0 - 1.1]$	0	12	23	25	26	23

Table 4.1: *Ratio between the final connectivity of the distributed solution and the centralized one for the 50 simulation runs. The adaptive case is indicated with $n_i(0)$. The cases $N \in \{20, 40\}$ correspond to a random feasible initial configuration (not necessarily a line).*

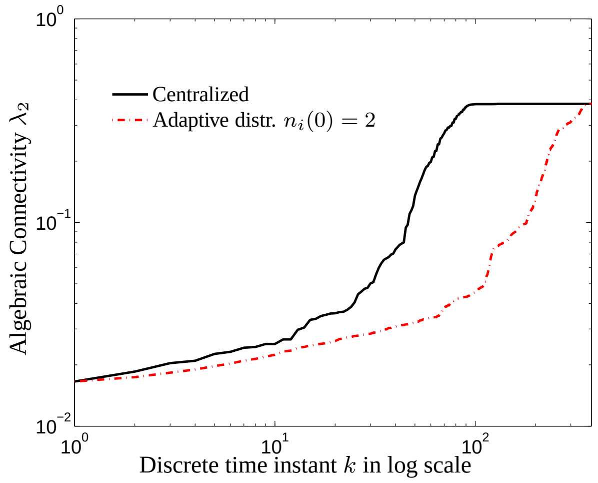


Figure 4.7: *Algebraic connectivity as a function of time k for both the centralized and the distributed solutions of Figure 4.8.*

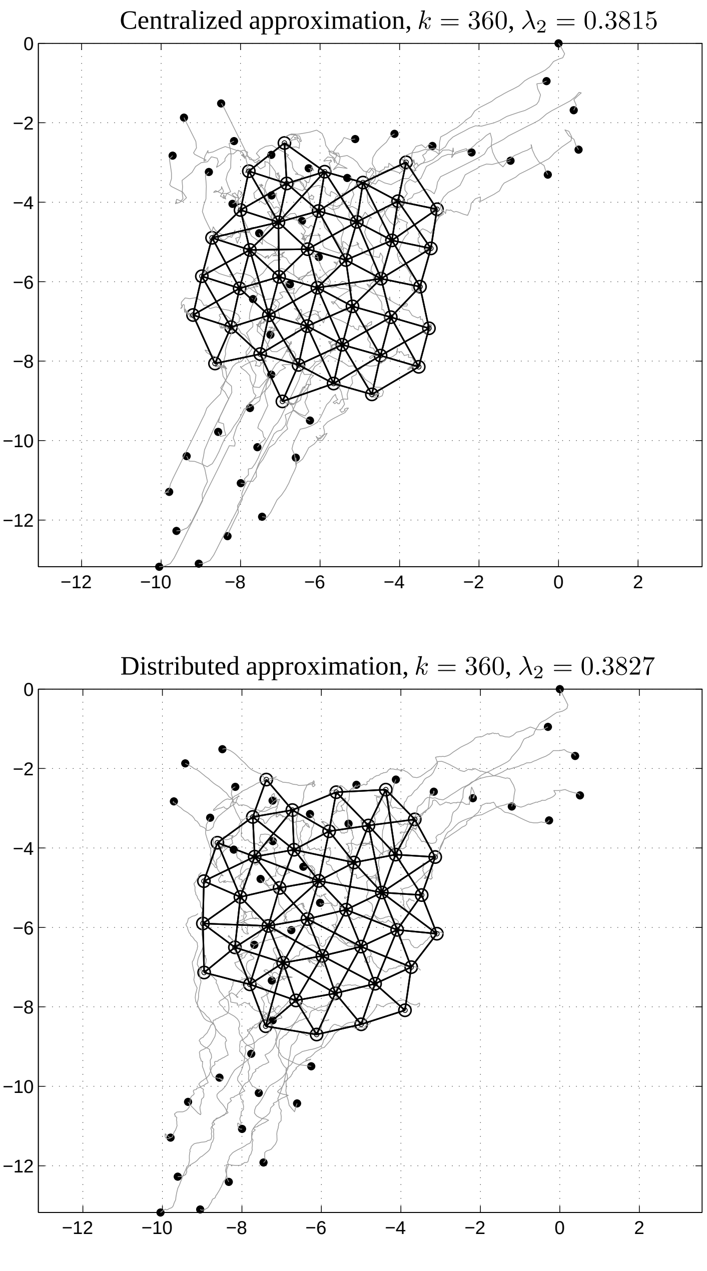


Figure 4.8: *Simulation results of the centralized and distributed approach (adaptive case). The initial positions are marked with black dots. The final positions are marked with circles. The bold lines represent the final communication graph and the thin lines the agent trajectories.*

4.3 Multi-Target Tracking

In this section we extend the proposed distributed solution for the maximization of the algebraic connectivity of the communication graph of a group of moving robots (Algorithm 4.1), to be able to tackle a multi-target tracking problem. In particular we will formulate the problem as the joint optimization of the connectivity among the agents and the number of targets in view.

We remark that in this section we will refer to target tracking as the problem of positioning the robots in order to ensure that the targets are in the detection range of the robots' sensors. Sometimes we will denote this detection range as sensing range, or visual range. We remark that the robots are not estimating the positions of the targets, but these are supposed to be known once the targets are in the detection range.

The works of (Grocholsky et al., 2003, Spletzer and Taylor, 2003, Chung et al., 2004, Martínez and Bullo, 2006, Olfati-Saber, 2007b, Simonetto et al., 2008, Zhou and Roumeliotis, 2008) provide a comprehensive overview of the multi-target tracking problem. Typical approaches consider a cost function based on the Fisher Information Matrix in order to determine robot movements that lead to an increase in the targets' visibility. However, even for a single target, the resulting optimization problem is nonlinear and NP-hard (Zhou and Roumeliotis, 2008). As a result, several alternative formulations relying on potential fields, gradient-descent, Monte Carlo methods, and linear approximations have been proposed, by sacrificing robot connectivity / target visibility guarantees, generality of the framework, or real-time applicability. Recently, an approximate formulation of the problem has been suggested using Semi-Definite Programming (Derenick et al., 2009, 2010), which is based on the tools of (Kim and Mesbahi, 2006, Boyd, 2006). Contrary to the aforementioned literature, this framework allows both the connectivity of the robotic network and the visibility of the targets to be considered *simultaneously*, in the same optimization problem. This is also the framework we will use in proposing our distributed solution.

4.3.1 Problem Formulation

We consider a group of M moving targets, in addition to the N mobile agents. We denote with q the index of the target. We consider both the agents and the targets to live on a two-dimensional plane, while for simplicity of exposition, in this section as in (Kim and Mesbahi, 2006, Derenick et al., 2009), we assume discrete-time agent dynamics of the form

$$x_i(k+1) = x_i(k) + v_i(k)\Delta t, \qquad i = 1, \dots, N \tag{4.50}$$

where $v_i(k)$ is the velocity control input and Δt the sampling time. We assume

$$||v_i(k)|| \leq v_{\max,i}.$$

Let $x(k) \in \mathbb{R}^{2N}$ be the collection of the agents' positions, i.e., $x(k) = (x_1^\top(k), \dots, x_N^\top(k))^\top$ and let $z_q(k) \in \mathbb{R}^2$ be the position of the q-th target at time k, while $z(k) = (z_1^\top(k), \dots, z_M^\top(k))^\top$ defines the collection of the targets' positions. We assume that the agents know their own position and the position of the targets they can detect, and that they have computation and communication capability onboard. We assume the targets can be represented as discrete-time dynamical systems

$$z_q(k+1) = z_q(k) + \mathbf{w}_q(k)\Delta t, \qquad q = 1, \ldots, M \tag{4.51}$$

where $\mathbf{w}_q(k) \in \mathbb{R}^2$ is a bounded input term, i.e. $||\mathbf{w}_q(k)|| \leq \mathbf{w}_{\max,q}$. The set of reachable positions $\mathcal{Z}_q(k)$ at time k, is the disc centered at the previous known position $z_q(k-1)$ with radius $\mathbf{w}_{\max,q}\Delta t$. We make the following assumption:

Assumption 4.4 (Slow targets) *The maximum target velocity is less than the agents' maximum velocity, i.e.,* $\mathbf{w}_{\max,q} < v_{\max,i}$ *for all pairs* (q, i).

This assumption has the scope to limit the targets' velocity, thus it has the scope to eliminate situations in which the agents would not possibly track them.

We model the communication graph among the agents $\mathcal{G}$ in the same way as done in Section 4.2, as well as the neighborhood sets $\mathcal{N}_i$ and $\mathcal{N}_i^+$. We define the collection of agents that are within their detection range of target q as $\mathcal{R}_q$[6]. These are considered as the neighboring agents of target q. The cardinality of $\mathcal{R}_q$, denoted as $|\mathcal{R}_q|$ expresses how many agents can detect a particular target. We introduce the following assumptions:

Assumption 4.5 (Initial feasibility) *At the initial time, i.e., at* $k = 0$, *the communication graph* $\mathcal{G}$ *is connected and each target* q *is detected by at least an agent, i.e.,* $|\mathcal{R}_q| > 0$ *for all* q.

Assumption 4.6 (Well-posedness) *At any time* $k > 0$, *there exist agent positions* $x(k+1)$, *independent of* $x(k)$, *which guarantee that the communication graph* $\mathcal{G}$ *remains connected and* $|\mathcal{R}_q| > 0$ *for each target* q.

This last assumption ensures that the problem is well-posed, but it does not guarantee feasibility at each time step. In fact, $x(k+1)$ depends on $x(k)$ via the dynamical equation (4.50), therefore the $x(k+1)$ provided by the assumption could be unreachable, given the current position $x(k)$. In practice, Assumption 4.6 requires that the targets do not move arbitrarily far away from each other compromising the connectedness of the communication graph.

We use the weights $0 \leq \mathrm{v}_{qi} \leq 1$ to model the link between target q and agent i, if they fall within the detection range (in a similar way as did for the Laplacian of the communication graph). The weights v_{qi} are also assumed to depend on the physical distance between the agent and target according to

$$\mathrm{v}_{qi}(k) = f_V(||z_q(k) - x_i(k)||^2), \tag{4.52}$$

where $f_V : \mathbb{R}_+ \to [0, 1]$ is a smooth nonlinear function with compact support. As for the case of communication weights, we assume that

$$\begin{aligned} f_V(||z_q(k) - x_i(k)||^2) &= 1, \text{ for } ||z_q(k) - x_i(k)||^2 \leq \rho_{1,\mathrm{v}} \\ f_V(||z_q(k) - x_i(k)||^2) &= 0, \text{ for } ||z_q(k) - x_i(k)||^2 \geq \rho_{2,\mathrm{v}}, \end{aligned}$$

[6] Formally we should write $\mathcal{R}_q(k)$, but we drop the dependency on k in order to simplify the notation.

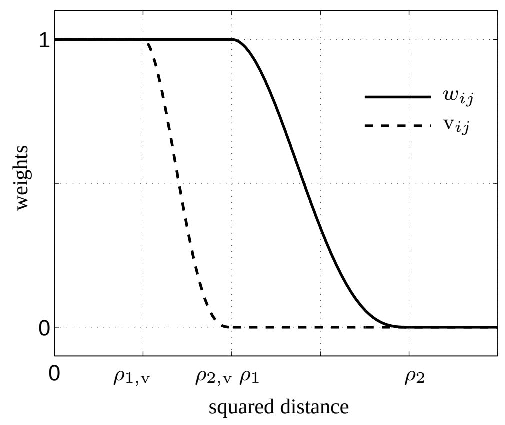


Figure 4.9: *The weighting functions.*

where $\rho_{1,\mathrm{v}}$ and $\rho_{2,\mathrm{v}}$ are positive scalars.

In Figure 4.9 we report a qualitative representation of the function f_V as well as the weighting function for the communication weights f_w as expressed in (4.4).

We assume:

Assumption 4.7 *Each agent that can detect a target is assumed to be able to communicate with all other agents that detect the same target, i.e.,* $2\rho_{2,\mathrm{v}} < \rho_2$.

This assumption will be instrumental for the proposed distributed solution in Section 4.3.3.

To characterize how a target is connected to agents we introduce the sum of the detection weights $\mathrm{v}_{qi}(k)$ of a target q as

$$\mathrm{v}_q(k) = \sum_{i \in \mathcal{R}_q} \mathrm{v}_{qi}(k) = \sum_{i \in \mathcal{R}_q} f_V(||z_q(k) - x_i(k)||^2), \tag{4.53}$$

and we note that if $\mathrm{v}_q(k) > n$ then $|\mathcal{R}_q| > n$, and therefore the target q is seen by at least n agents at the discrete time step k.

We are interested in maximizing visibility of the targets *and* maximizing communication connectivity among the agents. This can be posed as the joint maximization of the algebraic connectivity of the communication graphs and the sums of the detection weights by moving the agents into appropriate positions. This goal can be formulated in each discrete time step k as the following optimization problem, (Derenick et al., 2009, Boyd et al., 2006).

Problem 4.2 Multi-Target Tracking Problem

$$\mathbf{P}: \quad \underset{x(k+1),\gamma(k+1),\{\nu_1(k+1),\dots,\nu_N(k+1)\}}{\textbf{maximize}} \quad \alpha\gamma(k+1) + \sum_{q=1}^{M} \beta_q \nu_q(k+1) \tag{4.54a}$$

subject to

$$\gamma(k+1) \geq 0, \nu_q(k+1) \geq 0 \qquad q = 1,\dots,M \tag{4.54b}$$

$$L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma(k+1)I_N \tag{4.54c}$$

$$f_d(x_i(k+1), x_j(k+1)) > \rho_1, \quad \forall (i,j) \in \mathcal{E} \tag{4.54d}$$

$$\sum_{i\in\mathcal{R}_q} f_V(||z_q(k+1) - x_i(k+1)||^2) > \nu_q(k+1), \tag{4.54e}$$

$$\textit{for all } z_q(k+1) \in \mathcal{Z}_q(k+1),\ q = 1,\dots,M$$

$$||x_i(k+1) - x_i(k)|| \leq v_{\max,i}\Delta t \quad i = 1,\dots,N \tag{4.54f}$$

The constraints of problem (4.54) represent

- The connectivity of the communication graph (constraint (4.54c)),
- A minimal distance constraint, as in (4.8d), (constraint (4.54d)),
- The target detection constraint for all future reachable positions of the targets (constraint (4.54e)),
- The physical limitations of the agents' dynamics (constraint (4.54f)).

The decision variables of problem (4.54) are the agents' locations and the values of $\gamma(k+1)$, $\nu_q(k+1)$'s. Here the constants $\alpha \geq 0$ and $\beta_q \geq 0$, $q = 1,\dots,M$ model the scaled relative weights on the maximization goals. When one selects $\alpha = 0$, as in (Derenick et al., 2009), the problem becomes the maximization of detection connectivity with the targets while guaranteeing that the communication graph remains connected.

Remark 4.5 *We remark that if one substitutes the strict constraints:*

$$L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma(k+1)I_N$$

with the non-strictly positive ones:

$$L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succeq \gamma(k+1)I_N$$

the problem (4.54) could represent situations in which agents are allowed to form disconnected groups to follow different targets. In fact, if we allow the algebraic connectivity to become zero, i.e., $\lambda_2(L) = 0$, by the non-strict inequality, we implicitly allow the graph to become disconnected in order to better track the targets.

However, it is important to notice that extra care has to be put in the case of non-strict inequalities in the numerical method used. See for example (Boyd et al., 1994) for some details.

4.3.2 Centralized Approach

Convex Approximation

Problem (4.54) is non-convex given that we are optimizing over the positions x and the entries of the Laplacians are nonlinear functions of x. We employ the same standard convex approximation to this problem formulation, as done in Section 4.2, and we define

$$\triangle \mathrm{v}_q(z_q(k+1), x(k+1)) := \sum_{i\in\mathcal{R}_q} \left(\mathrm{v}_{qi}(k) + {c^{\mathrm{v}}_{qi}}^{\top}(\delta x_i(k+1) - \delta z_q(k+1))\right).$$

where

$$c^{\mathrm{v}}_{qi} = \left.\frac{\partial f_V}{\partial d^2_{ij}}\frac{\partial d^2_{ij}}{\partial x_i}\right|_{x_i(k),z_q(k)} = -\left.\frac{\partial f_V}{\partial d^2_{ij}}\frac{\partial d^2_{ij}}{\partial z_q}\right|_{x_i(k),z_q(k)}. \tag{4.55}$$

This allows us to formulate the following convex approximation of the problem (4.54):

$$\triangle \mathbf{P}(x(k+1), z(k+1), \rho_1, v_{\max,i}):$$

$$\begin{aligned}
&\underset{x(k+1),\gamma(k+1),\{\nu_1(k+1),\ldots,\nu_N(k+1)\}}{\textbf{maximize}} \quad \alpha\gamma(k+1) + \sum_{q=1}^{M}\beta_q\nu_q(k+1) && \text{(4.56a)}\\
&\textbf{subject to}\\
&\gamma(k+1)\geq 0, \nu_q(k+1)\geq 0 \qquad q=1,\ldots,M && \text{(4.56b)}\\
&\triangle L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^T \succ \gamma(k+1)I_N && \text{(4.56c)}\\
&\triangle f_d(x_i(k+1), x_j(k+1)) > \rho_1 \quad \forall (i,j)\in\mathcal{E} && \text{(4.56d)}\\
&\triangle \mathrm{v}_q(z^*_q(k+1), x(k+1)) > \nu_q(k+1) \qquad q=1,\ldots,M && \text{(4.56e)}\\
&||x_i(k+1) - x_i(k)|| \leq v_{\max,i}\Delta t \qquad i=1,\ldots,N && \text{(4.56f)}
\end{aligned}$$

where $z^*_q(k+1)$ is the *worst case* $z_q(k+1)$, which due to the linearity of the scalar inequality (4.56e) can be computed analytically (see Remark 4.6).

Remark 4.6 *The linearized version of constraint* (4.54e) *can be written as*

$$\sum_{i\in\mathcal{R}_q}\left(\mathrm{v}_{qi}(k) + {c^{\mathrm{v}}_{qi}}^{\top}(\delta x_i(k+1) - \delta z_q(k+1))\right) > \nu_q(k+1), \quad \forall z_q(k+1)\in\mathcal{Z}_q(k+1)$$

We can compute the worst case $z^*_q(k+1)$ as the one that minimizes

$$-\sum_{j\in\mathcal{R}_q} {c^{\mathrm{v}}_{qj}}^{\top}\delta z_q(k+1)$$

by solving the optimization problem

$$z^*_q(k+1) = \mathbf{arg}\min_{z_q(k+1)\in\mathcal{Z}_q(k+1)} -\sum_{j\in\mathcal{R}_q}{c^{\mathrm{v}}_{qj}}^{\top}\delta z_q(k+1)$$

This problem can be solved analytically, resulting in

$$z_q^*(k+1) = z_q(k) - [c_{(2)}^{\bar{v}} - c_{(1)}^{\bar{v}}]^\top \mathbf{w}_{\max,q} \Delta t$$

where $c_{(1)}^{\bar{v}}$ and $c_{(2)}^{\bar{v}}$ are the components of the normalized vector $\sum_{j \in \mathcal{R}_q} c_{qj}^{\mathrm{v}}$.

Properties of the Centralized Solution

To analyze the properties of the centralized solution we define:

Definition 4.4 *The decrease of the detection quality due to the targets' motion $\nu_q^-(k+1)$ is defined as*

$$\nu_q^-(k+1) := \nu_q(k) - \sum_{j \in \mathcal{N}_q} {c_{qj}^{\mathrm{v}}}^\top \delta z_q^*(k+1)$$

We note that $\nu_q^-(k+1) \geq 0$ by the definition of the weighting functions, therefore

$$\nu_q(k) \geq \sum_{j \in \mathcal{N}_q} {c_{qj}^{\mathrm{v}}}^\top \delta z_q^*(k+1).$$

The cost function of problem (4.56) at each time step k satisfies:

$$\alpha\gamma(k+1) + \sum_{q=1}^{M} \beta_q \nu_q(k+1) \geq \alpha\gamma(k) + \sum_{q=1}^{M} \beta_q \nu_q^-(k+1) \tag{4.57}$$

which indicates that the agents move in a way that improves the cost function if we consider only the current target locations. This inequality also implies that when the targets are stationary, the cost function is monotonically increasing.

The optimization problem that has been described in this section provides a framework to approach the joint connectivity and detection maximization problem in a centralized manner using linearization. In the following, we extend the distributed approach presented in Section 4.2 in order to allows the multi-target tracking problem (4.56) to be solved in a distributed fashion.

Remark 4.7 *The task to ensure persistent feasibility of the sequence of semi-definite programs (4.56) is not trivial for mobile targets. If the targets are stationary, persistent feasibility follows from the fact that the solution $x(k+1) = x(k)$ is feasible at the discrete time k, just as in Section 4.2. If the targets move arbitrarily, this property is instead difficult to impose. We believe that using the slow-target assumption (Assumption 4.4) one could derive conditions for persistent feasibility to hold. We leave this analysis for future research.*

4.3.3 Distributed Solution

In this section we present a non-iterative distributed solution to solve (4.56). By non-iterative we mean here that we will use only one round of communication/computation

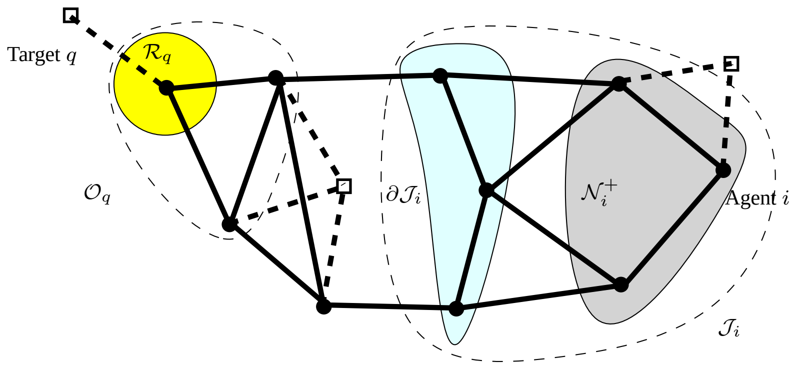


Figure 4.10: *Notation for the distributed solution. The targets are represented via squares, while the agents are circles. Communication links are shown via solid lines and detection links via dashed lines.*

among the different agents per optimization step k. Before presenting the main contribution of this section, we first introduce some notation and definitions. We then proceed to describe our non-iterative distributed solution method and its properties.

We will use the notation and definitions of Section 4.2.4. In addition, we consider the enlarged neighborhood set $\mathcal{J}_i$ to be of size 2, i.e., $n_i = 2$, for each agent. This implies that $\mathcal{J}_i^* = \mathcal{J}_i$, which will simplify the analysis of the distributed solution. We recall that the notation $x_{\mathcal{J}_i}$ denotes the vector containing all the positions of the agents $\mathcal{J}_i$ (where the order is not important). Since n_i is the same for all the agents, we will simplify the notation L_{i,n_i} with L_i and $\mathcal{E}_{i,n_i}$ with $\mathcal{E}_i$.

We will assume that agent i is aware of the targets it can detect directly and also the ones his first-order neighbors can detect. We will denote with $\mathcal{T}_i$ the set of all the targets that agent i is aware of. Correspondingly, we denote the vector containing all the positions of the targets in the set $\mathcal{T}_i$ with $z_{\mathcal{T}_i}$ (where the order is not important). Similarly to the enlarged neighborhood set for the agents we introduce the enlarged neighborhood set for the targets, indicating which agents are aware of a specific target q:

$$\mathcal{O}_q = \bigcup_{i \in \mathcal{R}_q} \mathcal{N}_i^+, \qquad q = 1, \ldots, M \tag{4.58}$$

whose cardinality is O_q. We note that these neighborhood sets contain only agents, and thus the maximum allowed cardinality is N. Figure 4.10 provides a graphical illustration of this notation. We also introduce a scaled maximum velocity $\tilde{v}_{\max,i}$ defined as

$$\tilde{v}_{\max,i} = \frac{N}{J_i} v_{\max,i}, \quad i = 1, \ldots, N \tag{4.59}$$

whose value varies from agent to agent. This quantity will be used to change the local constraints in such a way that the global solution constructed from the local ones satisfies the original constraint (4.56f), exactly as in Section 4.2.4 with the modified control constraints $\tilde{\mathcal{U}}_i$.

Our algorithm consists of two phases. First, each agent solves problem $\triangle\mathbf{P}_i$ defined as

$$\triangle\mathbf{P}_i(x_{\mathcal{J}_i}(k), z_{\mathcal{T}_i}(k), \tilde{\rho}_{1ij}, \tilde{v}_{\max,i}) :$$

$$\underset{x_{\mathcal{J}_i}(k+1),\gamma_i(k+1),\{\nu_q(k+1)\}_{q\in\mathcal{T}_i}}{\textbf{maximize}} \quad \alpha\gamma_i(k+1) + \sum_{q\in\mathcal{T}_i} \beta_q\nu_q(k+1) \tag{4.60a}$$

$$\textbf{subject to}$$

$$\gamma_i(k+1) \geq 0, \nu_q(k+1) \geq 0 \qquad q \in \mathcal{T}_i \tag{4.60b}$$

$$\triangle L_i(x_{\mathcal{J}_i}(k+1)) + \mathbf{1}_{J_i}\mathbf{1}_{J_i}^T \succ \gamma_i(k+1)I_{J_i} \tag{4.60c}$$

$$\triangle f_d(x_i(k+1), x_j(k+1)) > \tilde{\rho}_{1ij} \qquad \forall(i,j) \in \mathcal{E}_i \tag{4.60d}$$

$$\triangle \mathrm{v}_q(z_q^*(k+1), x_j(k+1)) > \nu_q(k+1) \qquad q \in \mathcal{T}_i, j \in \mathcal{J}_i \tag{4.60e}$$

$$||x_j(k+1) - x_j(k)|| \leq v_{\max,j}\Delta t \qquad j \in \mathcal{J}_i \tag{4.60f}$$

$$x_j(k+1) = x_j(k), \qquad \text{for } j \in \partial\mathcal{J}_i \tag{4.60g}$$

computing the solution $\tilde{x}_{\mathcal{J}_i}(k+1)$, which is composed of $\tilde{x}_{ij}(k+1)$ for each $j \in \mathcal{J}_i$. Thus, we will call $\tilde{x}_{ij}(k+1)$ the position of agent j as computed by agent i.

As a second phase, the solutions $\tilde{x}_{\mathcal{J}_i}(k+1)$ are shared within the enlarged neighborhood $\mathcal{J}_i$ and averaged according to

$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{J}_i} \frac{1}{N}\delta\tilde{x}_{ji}(k+1), \qquad i = 1, \dots, N \tag{4.61}$$

We note that the average (4.61) is a particular instance of the linear combination (4.30) with $s_i = 1/N$ and $n_i = 2$ for all the agents i. We remark that this particular choice will be important for the following analysis. Finally, we note that $\tilde{\rho}_{1ij}$ in (4.60d) is computed using (4.34) with $s_i = 1/N$ for all i.

Algorithm 4.2 summarizes the method. We note that steps 3-5 are implemented only once between subsequent robot movements, which makes the algorithm non-iterative.

We claim that *if we consider the global position vector* $x(k+1) = (x_1^\top(k+1), \dots, x_N^\top(k+1))^\top$ *resulting from* (4.61), then

(C1) The algebraic connectivity of the corresponding global linearized Laplacian $\triangle L(x(k+1))$ and $\triangle \mathrm{v}_q(z_q^*(k+1), x(k+1))$ are strictly positive;

(C2) All the constraints of the global problem are met.

Furthermore we claim that, as in the centralized approach:

(C3) The improvement property (4.57) remains valid for the cost function of $\triangle\mathbf{P}$ (4.56), when $x(k+1)$ comes from the distributed solution. Moreover, $\triangle\mathbf{P}$ is monotonically increasing when the targets are stationary.

We will prove these claims in three steps: Theorems 4.8, 4.9, and 4.10 establish (C1), by linking the average value (4.61) and the algebraic connectivity through the linear dependence of the linearized Laplacians on x. The constraint (4.60g) plays a crucial role here

Algorithm 4.2 Distributed Multi-Target Tracking

```
1: Input: $x_{\mathcal{J}_i}(k), z_{\mathcal{T}_i}(k)$
   ▷ Available data:   $f_w, f_d, f_V, \mathcal{J}_i, \mathcal{T}_i, (v_{\max,j} | \text{for } j \in \mathcal{J}_i), (\mathbf{w}_{\max,q} | \text{for } q \in \mathcal{T}_i), \rho_1$
2: Determine $c^{\mathrm{v}}_{qj}$ as (4.55) and $z^*_{\mathcal{T}_i}(k+1)$ as
       $z^*_q(k+1) = z_q(k) - [c^{\bar{\mathrm{v}}}_{(2)} - c^{\bar{\mathrm{v}}}_{(1)}]^\top \mathbf{w}_{\max,q} \Delta t, \quad q \in \mathcal{T}_i$
   where $c^{\bar{\mathrm{v}}}_{(1)}$ and $c^{\bar{\mathrm{v}}}_{(2)}$ are the normalized components of $\sum_{j \in \mathcal{N}_q} c^{\mathrm{v}}_{qj}$
3: Solve $\Delta \mathbf{P}_i$ in (4.60) computing $\tilde{x}_{\mathcal{J}_i}(k+1)$ as
       $\triangle \mathbf{P}_i(x_{\mathcal{J}_i}(k), z_{\mathcal{T}_i}(k), \tilde{\rho}_{1ij}, \tilde{v}_{\max,i})$ :
       $\underset{x_{\mathcal{J}_i}(k+1), \gamma_i(k+1), \{\nu_q(k+1)\}_{q \in \mathcal{T}_i}}{\textbf{maximize}} \quad \alpha \gamma_i(k+1) + \sum_{q \in \mathcal{T}_i} \beta_q \nu_q(k+1)$
         subject to
           $\gamma_i(k+1) \geq 0, \nu_q(k+1) \geq 0 \qquad q \in \mathcal{T}_i$
           $\triangle L_i(x_{\mathcal{J}_i}(k+1)) + \mathbf{1}_{J_i} \mathbf{1}^T_{J_i} \succ \gamma_i(k+1) I_{J_i}$
           $\triangle f_d(x_i(k+1), x_j(k+1)) > \tilde{\rho}_{1ij} \qquad \forall (i,j) \in \mathcal{E}_i$
           $\triangle \mathrm{v}_q(z^*_q(k+1), x_j(k+1)) > \nu_q(k+1) \qquad q \in \mathcal{T}_i, j \in \mathcal{J}_i$
           $||x_j(k+1) - x_j(k)|| \leq v_{\max,j} \Delta t \qquad j \in \mathcal{J}_i$
           $x_j(k+1) = x_j(k), \qquad \text{for } j \in \partial \mathcal{J}_i$
4: Communicate $\tilde{x}_{\mathcal{J}_i}(k)$ among members of $\mathcal{J}_i$
5: Average $x_i(k+1) = x_i(k) + \sum_{j \in \mathcal{J}_i} \frac{1}{N} \delta \tilde{x}_{ji}(k+1)$
6: Output: $x_i(k+1)$
```

to ensure the feasibility of the local solutions. Theorem 4.11 guarantees (C2), by showing how the scaled velocity (4.59) of the local problems ensure that the global solution, obtained via the average (4.61), satisfies the global constraints. Theorem 4.12 establishes (C3) by linking the variations of the local cost functions with the one of the global problem.

As in Section 4.2.4, consider the local problem $\triangle \mathbf{P}_i$ and its solution comprised of $\tilde{x}_{ij}(k+1)$ for all $j \in \mathcal{J}_i$. Recall the construction of the global vector $\tilde{x}^{(i)}(k+1)$ whose entries are determined based on the local solution as

$$\tilde{x}^{(i)}(k+1) = \begin{bmatrix} \tilde{x}^{(i)}_1(k+1) \\ \vdots \\ \tilde{x}^{(i)}_N(k+1) \end{bmatrix}, \quad j = 1, \ldots, N$$

$$\text{with } \tilde{x}^{(i)}_j(k+1) = \begin{cases} \tilde{x}_{ij}(k+1) & \text{if } j \in \mathcal{J}_i \\ x_j(k) & \text{otherwise} \end{cases} \tag{4.62}$$

where we keep those agent positions that have not been optimized fixed, and we update the rest from the solution of the local problem.

The following theorem is the adaptation of Theorem 4.3 for the Multi-Target tracking case.

Theorem 4.8 (C1.0) *The positions $\tilde{x}^{(i)}(k+1)$ in (4.62) constructed from the solution of* *the local problem* $\triangle \mathbf{P}_i$ *in* (4.60), *keep the global linearized Laplacian matrix connected:* $\triangle L(\tilde{x}^{(i)}(k+1)) + \mathbf{1}_N \mathbf{1}_N^\top \succ 0$.

Proof. At time step k, the graph $\mathcal{G}(x(k))$ is connected. We can divide this graph in two overlapping parts, $\mathcal{G}(x_{\mathcal{J}_i}(k))$, which is connected by definition, and $\mathcal{G}(x_{\neg\mathcal{J}_i}(k)) \cup \mathcal{G}(x_{\partial\mathcal{J}_i}(k))$, where with $\neg\mathcal{J}_i$ we indicate the collection of agents not present in $\mathcal{J}_i$. At time step k we know that:

1. $\mathcal{G}(x(k+1)) = \mathcal{G}(x_{\mathcal{J}_i}(k+1)) \cup (\mathcal{G}(x_{\neg\mathcal{J}_i}(k)) \cup \mathcal{G}(x_{\partial\mathcal{J}_i}(k)))$;
2. $\mathcal{G}(x_{\mathcal{J}_i}(k+1)) \cap (\mathcal{G}(x_{\neg\mathcal{J}_i}(k)) \cup \mathcal{G}(x_{\partial\mathcal{J}_i}(k))) = \mathcal{G}(x_{\partial\mathcal{J}_i}(k)) \neq \{\emptyset\}$;

where we use the definition of $x^{(i)}(k+1)$ in (4.62) and the constraint (4.60g) on the bordering agents. Noticing that $\mathcal{G}(x_{\mathcal{J}_i}(k+1))$ is also connected as imposed by the local optimization problem, the claim follows. □

The following lemma is a particular instance of Lemma 4.3.

Lemma 4.4 *The following equality holds:*

$$\triangle L(\delta x(k+1)) = \sum_{i=1}^{N} \frac{1}{N} \triangle L(\delta \tilde{x}^{(i)}(k+1)) \tag{4.63}$$

Proof. The proof follows from Lemma 4.3 with $s_i = 1/N$. □

Using Theorem 4.8 and Lemma 4.4 we can now prove the strict positiveness of the algebraic connectivity of the global linearized Laplacian $\triangle L(x(k+1))$, formally stated in Theorem 4.9.

Theorem 4.9 (C1.1) *The algebraic connectivity of the global linearized Laplacian* $\triangle L(x(k+1))$ *is strictly positive,* $\triangle L(x(k+1)) + \mathbf{1}_N \mathbf{1}_N^\top \succ 0$ *where* $x(k+1)$ *is computed by the average* (4.61).

Proof. Theorem 4.8 implies $(\triangle L(\tilde{x}^{(i)}(k+1)) + \mathbf{1}_N \mathbf{1}_N^\top)/N \succ 0$ for all i. Thus summing over all agents leads to

$$\sum_{i=1}^{N} \frac{1}{N} (\triangle L(\tilde{x}^{(i)}(k+1)) + \mathbf{1}_N \mathbf{1}_N^\top) \succ 0$$

or by linearity of $\triangle L(\tilde{x}^{(i)}(k+1))$ with respect to x, Equation (4.44),

$$(\triangle L(x(k)) + \mathbf{1}_N \mathbf{1}_N^\top) + \sum_{i=1}^{N} \frac{1}{N} \triangle L(\delta \tilde{x}^{(i)}(k+1)) \succ 0 \tag{4.64}$$

Considering the weighted sum $x_i(k+1)$ in (4.61), and the associated global vector $x(k+1)$, by Lemma 4.4 follows the desired property $\triangle L(x(k+1)) + \mathbf{1}_N \mathbf{1}_N^\top \succ 0$. □

Remark 4.8 *We note that the validity of Theorem 4.9 is limited to the case in which $\sum_{i=1}^{N} s_i = 1$, e.g., $s_i = 1/N$ for all the agents. In fact, the expression (4.64) can be generalized for the linear combination (4.30) in:*

$$\left(\sum_{i=1}^{N} s_i\right)(\triangle L(x(k)) + \mathbf{1}_N\mathbf{1}_N^\top) + \sum_{i=1}^{N} s_i \triangle L(\delta\tilde{x}^{(i)}(k+1)) \succ 0$$

that leads to $\triangle L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^\top \succ 0$ only if $\sum_{i=1}^{N} s_i = 1$.

Theorem 4.10 (C1.2) *The local constraint $\tilde{\nu}_q(k+1) \geq 0$ is a sufficient condition for all the targets to be connected at least to one agent, i.e., $\nu_q(k+1) \geq 0\,, \forall q = 1, \ldots, M$.*

Proof. We need to prove the implication

$$\tilde{\nu}_q(k+1) \geq 0 \Rightarrow \nu_q(k+1) \geq 0, \quad \text{for } q = 1, \ldots, M \tag{4.65}$$

We start re-interpreting the condition $\tilde{\nu}_q(k+1) \geq 0$. To this aim, consider target q, which appears in the local constraints of subproblem $\triangle\mathbf{P}_p$, $p \in \mathcal{O}_q$ as

$$\sum_{j\in\mathcal{R}_q} \triangle \mathrm{v}_{qj}(z_q^*(k+1), \tilde{x}_{pj}(k+1)) > \tilde{\nu}_{qp}(k+1) \geq \tilde{\nu}_q(k+1) \geq 0$$

for a suitable $\tilde{\nu}_q(k+1)$. This constraint can be written in the equivalent form

$$\sum_{j\in\mathcal{R}_q} \left({c_{qj}^{\mathrm{v}}}^\top \delta\tilde{x}_{pj} + c_j^\nu\right) > \tilde{\nu}_q(k+1) \geq 0 \tag{4.66}$$

where we have defined

$$0 \leq \sum_{j\in\mathcal{R}_q} c_j^\nu = \sum_{j\in\mathcal{R}_q} \mathrm{v}_{qj}(z_q(k), x_{pj}(k)) - \sum_{j\in\mathcal{R}_q} {c_{qj}^{\mathrm{v}}}^\top \delta z_q^*(k+1) = \nu_q^-(k+1)$$

and we note that due to Assumption 4.7, $\forall p \in \mathcal{O}_q$ we have $\mathcal{R}_q \subseteq \mathcal{O}_p$, therefore constraint (4.66) can indeed be computed locally. Starting from Equation (4.66), summing over the p's and dividing by N:

$$\sum_{p\in\mathcal{O}_q}\sum_{j\in\mathcal{R}_q} \left({c_{qj}^{\mathrm{v}}}^\top \frac{\delta\tilde{x}_{pj}(k+1)}{N} + \frac{c_j^\nu}{N}\right) > \sum_{p\in\mathcal{O}_q} \frac{\tilde{\nu}_q(k+1)}{N}$$

or

$$\sum_{p\in\mathcal{O}_q}\sum_{j\in\mathcal{R}_q} {c_{qj}^{\mathrm{v}}}^\top \frac{\delta\tilde{x}_{pj}(k+1)}{N} > \frac{O_q}{N}\tilde{\nu}_q(k+1) - \frac{O_q}{N}\sum_{j\in\mathcal{R}_q} c_j^\nu \tag{4.67}$$

We need to prove that globally

$$\sum_{j\in\mathcal{R}_q} \left({c_{qj}^{\mathrm{v}}}^\top \delta x_j(k+1) + c_j^\nu\right) > \nu_q(k+1) \geq 0 \tag{4.68}$$

which can be rewritten substituting the average (4.61) in (4.68) as

$$\sum_{j\in\mathcal{R}_q}\sum_{p\in\mathcal{J}_j}\left(c_{qj}^{\mathrm{v}\,\top}\frac{\delta\tilde{x}_{pj}(k+1)}{N}+c_j^{\nu}\right)>\nu_q(k+1). \tag{4.69}$$

However, since

$$\bigcup_{j\in\mathcal{R}_q}\mathcal{J}_j=\mathcal{O}_q\cup\left(\bigcup_{j\in\mathcal{R}_q}\partial\mathcal{J}_j\right)$$

and due to constraint (4.60g), then the expression (4.69) becomes

$$\sum_{j\in\mathcal{R}_q}\sum_{p\in\mathcal{O}_q}\left(c_{qj}^{\mathrm{v}\,\top}\frac{\delta\tilde{x}_{pj}(k+1)}{N}+c_j^{\nu}\right)>\nu_q(k+1),$$

or switching the sum operators

$$\sum_{p\in\mathcal{O}_q}\sum_{j\in\mathcal{R}_q}c_{qj}^{\mathrm{v}\,\top}\frac{\delta\tilde{x}_{pj}(k+1)}{N}>\nu_q(k+1)-O_q\sum_{j\in\mathcal{R}_q}c_j^{\nu}. \tag{4.70}$$

By direct comparison of expression (4.67) and (4.70), by the arbitrariness of the choice of $\nu_q(k+1)$, we can always pick

$$\frac{O_q}{N}\tilde{\nu}_q(k+1)+O_q\sum_{j\in\mathcal{R}_q}c_j^{\nu}-\frac{O_q}{N}\sum_{j\in\mathcal{R}_q}c_j^{\nu}=\nu_q(k+1)$$

for some $\nu_q(k+1)$. This leads to

$$\tilde{\nu}_q(k+1)+(N-1)\nu_q^{-}(k+1)=\frac{N}{O_q}\nu_q(k+1). \tag{4.71}$$

We can now prove the implication (4.65) by the relation (4.71). In fact, by assumption $\tilde{\nu}_q(k+1)\geq 0$, while we know that $\nu_q^{-}(k+1)\geq 0$ by definition. This implies by (4.71), that $\nu_q(k+1)\geq 0$. □

Theorem 4.11 (C2) *The global constraints* (4.56b)-(4.56f) *are satisfied by the average solution* (4.61).

Proof. The constraints (4.56b)-(4.56e) are ensured via Theorems 4.2, 4.9, and 4.10. Consider now the constraint (4.56f), for each subproblem we have

$$||\delta\tilde{x}_{ii}(k+1)||<\tilde{v}_{\max,i}=\frac{N}{J_i}v_{\max,i}$$

and for the global problem:

$$||\delta x_i(k+1)||<\sum_{j\in\mathcal{J}_i}\frac{1}{N}||\delta\tilde{x}_{ji}(k+1)||<v_{\max,i}$$

Thus $x(k+1)$ satisfies also (4.56f) and (C2) is established. $\square$

Similarly to the centralized case, the global cost function of $\triangle\mathbf{P}$ satisfies the improvement property (4.57), as formally stated in the following theorem.

Theorem 4.12 (C3) *The global cost function of $\triangle\mathbf{P}$ satisfies the following improvement property:*

$$\alpha\gamma(k+1) + \sum_{q=1}^{M} \beta_q \nu_q(k+1) \geq \alpha\gamma(k) + \sum_{q=1}^{M} \beta_q \nu_q^-(k+1) \tag{4.72}$$

where the solution at time $k+1$ is computed from the local problems with the average (4.61).

Proof. We start rewriting (4.72) in the equivalent semi-definite form:

$$\alpha\left(\triangle L(x(k+1)) + \mathbf{1}_N\mathbf{1}_N^\top\right) + I_N \sum_{q=1}^{M} \beta_q \nu_q(k+1) \succeq \alpha\left(\triangle L(x(k)) + \mathbf{1}_N\mathbf{1}_N^\top\right) + I_N \sum_{q=1}^{M} \beta_q \nu_q^-(k+1)$$

For optimality of the local problems, in each $\triangle\mathbf{P}_i$:

$$\alpha\left(\triangle L_i(\tilde{x}_{\mathcal{J}_i}(k+1)) + \mathbf{1}_{J_i}\mathbf{1}_{J_i}^\top\right) + I_{J_i} \sum_{q\in\mathcal{T}_i} \beta_q \tilde{\nu}_q(k+1) \succeq \alpha\left(\triangle L_i(\tilde{x}_{\mathcal{J}_i}(k)) + \mathbf{1}_{J_i}\mathbf{1}_{J_i}^\top\right) + I_{J_i} \sum_{q\in\mathcal{T}_i} \beta_q \nu_q^-(k+1)$$

or

$$\alpha\triangle L_i(\delta\tilde{x}_{\mathcal{J}_i}(k+1)) + I_{J_i} \sum_{q\in\mathcal{T}_i} \beta_q(\tilde{\nu}_q(k+1) - \nu_q^-(k+1)) \succeq 0$$

For constraint (4.60g) and Assumption 4.7, this can be written as

$$\alpha\triangle L_i(\delta\tilde{x}^{(i)}(k+1)) + I_N \sum_{q\in\mathcal{T}_i} \beta_q(\tilde{\nu}_q(k+1) - \nu_q^-(k+1)) \succeq 0$$

Summing over the agents and dividing by N

$$\alpha\sum_{i=1}^{N}\frac{1}{N}\triangle L_i(\delta\tilde{x}^{(i)}(k+1)) + I_N \sum_{i=1}^{N}\frac{1}{N}\sum_{q\in\mathcal{T}_i} \beta_q(\tilde{\nu}_q(k+1) - \nu_q^-(k+1)) \succeq 0$$

the second term can be written as:

$$\sum_{i=1}^{N}\frac{1}{N}\sum_{q\in\mathcal{T}_i} \beta_q(\tilde{\nu}_q(k+1) - \nu_q^-(k+1)) = \sum_{q=1}^{M}\frac{J_q}{N}\beta_q(\tilde{\nu}_q(k+1) - \nu_q^-(k+1))$$

by Theorem 4.9 and Eq. (4.71):

$$\alpha \triangle L_C(\delta\tilde{x}(k+1)) + I_N \sum_{q=1}^{M} \beta_q(\nu_q(k+1) - \nu_q^-(k+1)) \succeq 0$$

□

Corollary 4.1 (Stationary targets) *When the targets are stationary the global cost function of* $\triangle\mathbf{P}$ *using the average* (4.30) *is monotonically increasing.*

The proof is straightforward by noticing that in this case $\nu_q^-(k) = \nu_q(k)$.

4.3.4 Simulation Results

In this section we present simulation results comparing the centralized approximation with our distributed approach. We consider $N = 10$ agents and $M = 3$ targets. The agents lie initially close to the x-axis, while the targets start at $(0, 0)$. We consider $v_{\max,i} = 0.25$, $\mathbf{w}_{\max,q} = v_{\max,i}/15$, and $\Delta t = 1$. For the weighting function of Figure 4.9, we take $\rho_1 = 0.75$, $\rho_2 = 3$ for w_{ij} and $\rho_1 = 0.75$, $\rho_2 = 1.25$ for v_{qi}.

We select $\alpha = 1$ and $\beta_q = 10^4$ for all three targets. We drive the targets in opposite direction with a non zero-mean bounded noise process. Figures 4.11-4.12 show the trajectories of the agents/targets for both the centralized and the distributed solutions (Both the initial positions and the target trajectories are the same in the two simulations). Although the agents' trajectories are different in the two approximations, in both cases the agents maintain a certain level of connectivity (see Figure 4.13), while keeping track of the moving targets. The differences are due to the linearizations, which are trajectory dependent.

As in the maximization of the algebraic connectivity, the simulations illustrate that the centralized approximation is faster to respond than the distributed one, although both of them meet the constraints of the optimization problem.

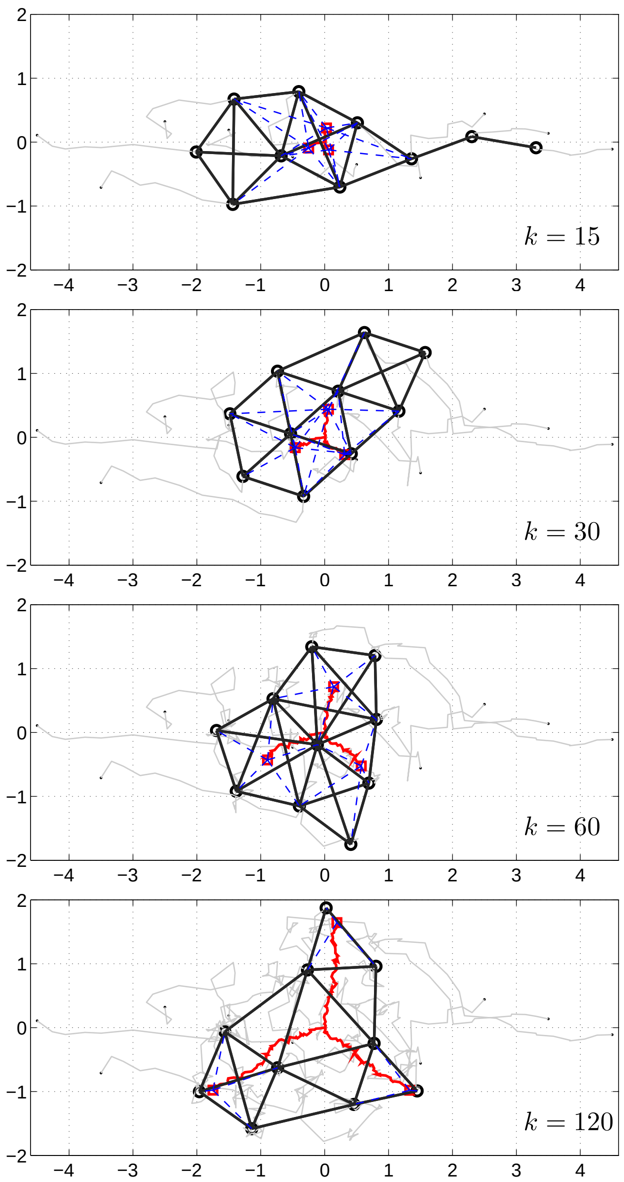


Figure 4.11: *Centralized approximations at four different discrete time instants. The thin black lines represent the agents' trajectories, which start from the points marked with small black dots. The targets' trajectories are in red and start from* $(0, 0)$*. Black circles represent the agents, squares the targets, solid lines are the communication links and dashed lines the detection links.*

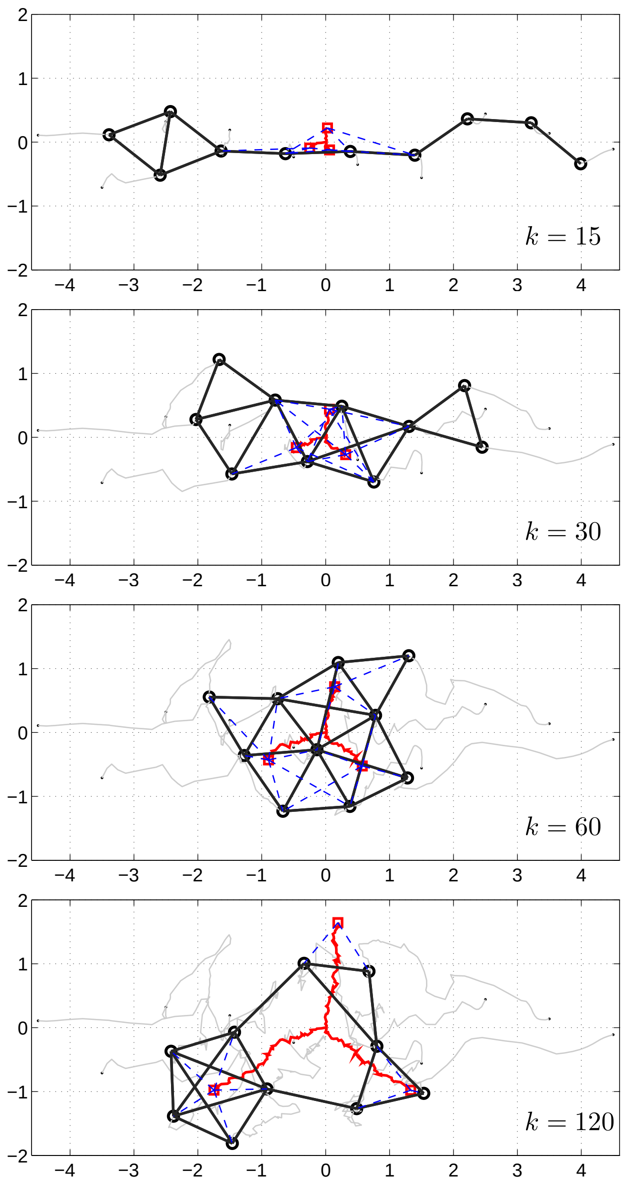


Figure 4.12: *Distributed approximations at four different discrete time instants. The thin black lines represent the agents' trajectories, which start from the points marked with small black dots. The targets' trajectories are in red and start from* $(0, 0)$. *Black circles represent the agents, squares the targets, solid lines are the communication links and dashed lines the detection links.*

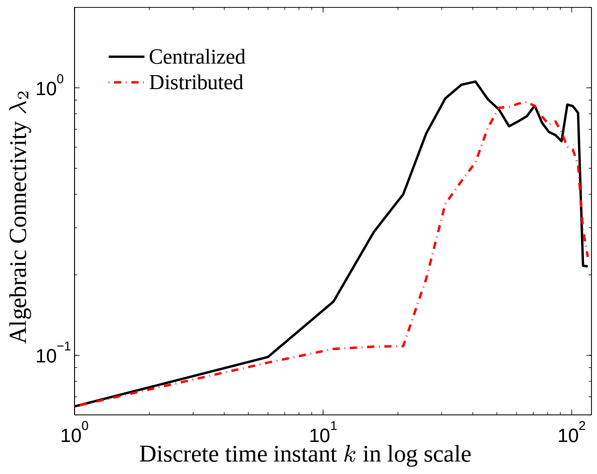


Figure 4.13: *Algebraic connectivity as a function of time k for both the centralized and the distributed solutions of Figure 4.12.*

4.4 Conclusions

In this chapter we have studied two distributed control problems for robotic networks. First, we have presented a distributed solution to the maximization of the algebraic connectivity of the communication graph in a robotic network. Our characterization can handle more realistic agent dynamics than the methods available in the literature and the resulting optimization problem is proven to be feasible at each time step under reasonable assumptions. Furthermore the solution can be adjusted based on available resources using local relative sub-optimality measures to aid in adapting the neighborhood size to the agents' needs. Simulation results confirm the efficacy of our distributed approach and show its practical applicability.

Second, based on and extending the techniques of the first part of the chapter, we have proposed a distributed and non-iterative solution for the problem of collectively tracking multiple mobile targets using a robotic network, while maintaining a certain level of connectivity.

4.5 Open Problems and Future Work

The presented problems and our proposed solutions are closely related, and therefore the following open research challenges apply to both.

Generalization of the Proposed Solutions

The first open problem we discuss is the generalization of the proposed algorithms in order to guarantee additional properties for the distributed solutions. In particular,

- Robustness of the proposed algorithms (Algorithm 4.1 and 4.2) against estimation errors. In particular, how to ensure connectivity when the positions of the nearby agents are known only with a level of uncertainty;
- Applicability of the proposed distributed schemes to a broader class of problem formulations involving LMI constraints of the form

  $$A(\mathbf{x}) \succ 0$$

  for a generic positive semi-definite matrix A with the same sparsity as the graph Laplacian. This would enable the possibility to solve in a distributed way a broader class of convex optimization problems.
- The possibility to further decrease the computation and communication effort (while still satisfying the constraints of the global problem (either (4.26) or (4.56)) by the use of gossip, randomized, or token-based methods (Boyd et al., 2006, Fagnani and Zampieri, 2008, Xu et al., 2008, Johansson et al., 2009). In this case each agent decides randomly to which neighbor to communicate. This idea has been briefly studied in (Simonetto et al., 2010b), but deserves more in-depth analysis.

Generalization of the Problem Formulation: agents as general LTI systems

Another interesting open problem is the generalization of the formulation to handle general LTI systems in the centralized problem (4.26). We start from a generalization of (4.16) considering the $(M+2)$-th order system:

$$\begin{pmatrix} x_i(\tau+1) \\ v_i(\tau+1) \\ y_{1i}(\tau+1) \\ \vdots \\ y_{Mi}(\tau+1) \end{pmatrix} = \begin{pmatrix} I_3 & \star & \star & \cdots \\ 0_3 & \star & \star & \cdots \\ 0_3 & \star & \star & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_i(\tau) \\ v_i(\tau) \\ y_{1i}(\tau) \\ \vdots \\ y_{Mi}(\tau) \end{pmatrix} + \begin{pmatrix} 0_3 \\ \vdots \\ b_{1i} I_3 \end{pmatrix} u_i(\tau)$$

where the y_{ji} are additional states, the stars represent non-zero elements, and $b_{1i} \in \mathbb{R}_0$. It is not difficult to see that Algorithm 4.1 is also applicable to these types of systems, under quite general assumptions and minor modifications. The key idea is to compute the control actions every $M+2$ steps while the crucial drawback is that the larger $M+2$ is, the more ρ_1 has to be shrunk to accommodate the collision avoidance requirement (see condition (4.25) which has to be generalized in this case in a straightforward manner).

Consider now the generic LTI system

$$\mathbf{x}_i(\tau+1) = A_i \mathbf{x}_i(\tau) + B_i u_i(\tau)$$

where the couple (A_i, B_i) is controllable and where the state can be partitioned as $(x_i(\tau)^\top, \xi_i(\tau)^\top)^\top$. In order to apply Algorithm 4.1, we need to characterize a modification of the set $\mathcal{F}_i$ which is defined as:

$$\mathcal{F}_i^{T_i} = \{\mathbf{x}_i(\tau) \in \mathbb{R}^{3(M+2)} | \exists u_i(\tau) \in \bar{\mathcal{U}}_i, \ldots, u_i(\tau+T-1) \in \bar{\mathcal{U}}_i \text{ such that}$$

$$A_i^{T_i}\mathbf{x}_i(\tau) + \sum_{h=0}^{T_i-1} A_i^{T_i-1-h} B_i u(\tau+h) = (\mathbf{x}_i(\tau)^\top, 0)^\top, \forall \tau \in \mathbb{N}^+$$

By computing $\mathcal{F}_i^{T_i}$, we can extend Algorithm 4.1 also to general LTI systems, calculating the control every $\max_i\{T_i\}$ time steps. However, several issues have to be addressed: *(i)* the parameter T_i is agent-dependent; *(ii)* the set $\mathcal{F}_i^{T_i}$ depends also on the position $x_i(\tau)$ restricting the area in which the agents can move; *(iii)* since $\max_i\{T_i\}$ can be in general quite large, the condition on ρ_1 could be rather limiting and it could conflict with the requirements on $\mathcal{F}_i^{T_i}$.

Generalization of the Problem Formulation: full-state dependent Laplacians

The third challenge we discuss is the generalization of the proposed methods to full-state dependent Laplacians, where the connectivity depends not only on the position of the agents, but also on other part of their state, e.g., their velocities. This could extend the validity of the approach to more complex scenario where, for example, the connectivity depends on the relative angles (limited field of view communication), or relative (angular) velocities.

An Improvement of the Linearization Procedure

Another open problem is the analysis of the implications of the linearized procedure of Section 4.2.1. This study is important in the case the positions of the agents vary significantly between two discrete time steps k. In particular, we are interested in cases in which $||x_i(k) - x_i(k-1)|| \gg 0$, which can occur if the constraints on the dynamics of the agents are not strict enough, or if no extra constraint is imposed which upper-bounds this difference.

Consider the centralized problem (4.26). Although, this can be shown to converge to a local maximum (Kim and Mesbahi, 2006), this property may be lost when the linearized variables (with $\triangle$'s) are significantly different from the actual ones (therefore when $||x_i(k) - x_i(k-1)||$ grows arbitrarily). This can be seen by a simple argument.

First of all, due to the convex dependency of $d_{ij}^2(x(k))$ on $x(k)$,

$$d_{ij}^2(k) \geq \triangle f_d(x_i(k), x_j(k)) \quad \text{for each } (i,j) \in \mathcal{E},$$

which is formalized in the following lemma.

Lemma 4.5 *The pairwise distance function satisfies* $d_{ij}^2(k) \geq \triangle f_d(x_i(k), x_j(k))$ *for each* $(i,j) \in \mathcal{E}$.

Proof. Direct calculation leads to

$$\begin{aligned} d_{ij}^2(k) &= ||x_i(k) - x_j(k)||^2 = ||x_i(k-1) + \delta x_i(k) - x_j(k-1) - \delta x_j(k)||^2 = \\ &= d_{ij}^2(k-1) + 2(x_i(k-1) - x_j(k-1))^\top(\delta x_i(k) - \delta x_j(k)) + \\ &\quad + ||\delta x_i(k) - \delta x_j(k)||^2 \end{aligned} \tag{4.73}$$

while:

$$\triangle f_d(x_i(k), x_j(k)) \quad = \quad d_{ij}^2(k-1) + 2(x_i(k-1) - x_j(k-1))^\top (\delta x_i(k) - \delta x_j(k)). \tag{4.74}$$

We need to show that $d_{ij}^2(k) \geq \triangle f_d(x_i(k), x_j(k))$; assuming that this is the case, substituting the expression (4.74), the following inequality should also hold

$$d_{ij}^2(k) \geq d_{ij}^2(k-1) + 2(x_i(k-1) - x_j(k-1))^\top (\delta x_i(k) - \delta x_j(k)),$$

or, substituting (4.73) for $d_{ij}^2(k)$,

$$||\delta x_i(k) - \delta x_j(k)||^2 \geq 0$$

which is a tautology. □

The fact that $d_{ij}^2(k) \geq \triangle f_d(x_i(k), x_j(k))$ for each $(i,j) \in \mathcal{E}$ leads to the possibility that $w_{ij}(k) \leq \triangle f_w(||x_i(k) - x_j(k)||^2)$. Since the eigenvalues of L depend on the weights, it is possible that λ_2 of $L(x(k))$, based on $d_{ij}^2(k)$, is smaller than the optimal $\gamma(k)$. This implies that we cannot guarantee the sequence of optimal $\gamma(k)$'s to be increasing and thus an important assumption of the proof in (Kim and Mesbahi, 2006) is not satisfied.

From an implementation perspective, one could think of using the following iterative procedure to re-ensure convergence. Consider the case $w_{ij}(k) < \triangle f_w(||x_i(k) - x_j(k)||^2)$ for some couple (i,j), then

1. Evaluate the partial derivatives c_{ij}^w and c_{ij}^d in the expansion of $w_{ij}(k)$ and $d_{ij}^2(k)$ (Equations (4.3)-(4.4)) at the averaged position $\bar{x} = (x(k) + x(k-1))/2$;
2. Recompute the solution of the problem (4.26);
3. Iterate until for all couples (i,j): $w_{ij} \geq \triangle f_w(||x_i(k) - x_j(k)||^2)$.

The evaluation of this scheme is another open research problem.

Multi-Target Tracking problem

Finally, for the multi-target tracking problem of Section 4.3, we can sketch two other interesting questions:

- How to ensure target-to-agent minimal-distance separation. This constraint could be posed as done in the agent-to-agent minimal-distance separation, but in the target case, the targets' future locations are unknown and therefore a more in depth study has to be performed;
- How to guarantee persistent feasibility for the sequence of Semi-Definite Programs (4.56) (Remark 4.7). If the targets are stationary, persistent feasibility follows from the fact that the solution $x(k+1) = x(k)$ is feasible at the discrete time k, just as in Section 4.2. If the targets move arbitrarily, this property is instead difficult to impose. We believe that using the slow-target assumption (Assumption 4.4) one could derive conditions for persistent feasibility to hold.

Chapter 5

Distributed Optimization Methods in Robotic Network Applications

Abstract — In this chapter we focus on convex and non-convex networked optimization problems with resource allocation constraints, which can be used in realistic robotic network applications where the mobile or non-moving devices share the same resources across the whole network.

First, we propose a regularized saddle-point algorithm for convex networked optimization problems with resource allocation constraints. Standard subgradient methods suffer from slow convergence and require excessive communication when applied to problems of this type. Our approach offers an alternative way to address these problems, and ensures that each iterative update step satisfies the resource allocation constraints. We derive step-size conditions under which the distributed algorithm converges geometrically to the regularized optimal value, and show how these conditions are affected by the underlying network topology. We illustrate our method on a robotic network application example where a group of mobile agents strive to maintain a moving target in the barycenter of their positions.

Second, we focus on a particular non-convex networked resource allocation problem, known as the Maximum Variance Unfolding problem and its dual, the Fastest Mixing Markov Process problem. These problems are of relevance for sensor networks and mobile robot applications. We solve both these problems with the same distributed primal-dual subgradient iterations whose convergence is proven even in the case of approximation errors in the calculation of the subgradients. Furthermore, we illustrate the importance of the algorithm for sensor network applications, among which localization problems, and we discuss some extensions to mobile robotic networks and to dispersion problems.

5.1 Introduction

In the previous chapters we have encountered and made use of distributed optimization problem formulations for different scenarios linked to fixed or mobile robotic networks. In most of the analyzed cases, the optimization framework was used to formulate application-specific problems. In this chapter, we shift our focus from these application-specific optimization problems to more abstract ones. In particular we study certain convex and non-convex optimization problems with resource allocation constraints, i.e., problems in which the decision variables are coupled by the allocation of a given resource (e.g., their sum has to be equal to a given constant). These problems can be thought of as a generalization of open research problems in many fields, among which sensor and robotic networks, as we will clearly illustrate in this chapter. This link is also one of the main reasons of our interest.

In Section 5.2 we focus on distributed convex problems. In particular we study primal-dual (saddle-point) iterative methods for solving convex optimization problems with resource allocation constraints in addition to convex constraints on local variables and sparse (convex) coupling constraints. Standard (sub)-gradient methods (Kiwiel, 2004, Nedić and Ozdaglar, 2009) have gained increased attention in the past years, yet their convergence rate is known to be rather low especially when the cost function is non-strictly convex. A recently proposed method (Devolder et al., 2011, Koshal et al., 2011) regularizes the initial convex problem and thereby increases the convergence rate of common algorithms delivering a solution arbitrarily close to the one of the original problem. Motivated by this strategy, we also make use of regularization and solve the resulting strictly convex problem via a saddle-point method. Furthermore, we incorporate the resource allocation equality constraints directly into the saddle-point iterations by extending the results of (Xiao and Boyd, 2006) (originally proposed for unconstrained problems). We derive step-size conditions that guarantee convergence of our iterative scheme, and show how these results are linked to the problem characteristics and the graph topology, respectively. In standard dual decomposition approaches (Johansson, 2008) one would typically dualize the equality constraints and use consensus mechanisms to distribute the resulting Lagrangian function over the network. This technique results in a slow convergence rate, especially if the network is sparsely connected. Our proposed approach incorporates the resource allocation constraints directly in the saddle-point iteration, and uses regularization to obtain faster convergence. This leads to an inherently distributed method, which converges to the solution of the original regularized problem.

Finally, we illustrate our algorithm on a realistic robotic network example where a number of mobile robots strive to keep a moving target in the barycenter of their positions. This scenario is motivated by our interest in robotic networks and the recent works in target tracking and target circumnavigation, e.g. Derenick et al. (2009), Shames et al. (2012), where distributed algorithms are required to be applicable in real-time.

In Section 5.3 we focus on a particular non-convex resource allocation problem, known as Maximum Variance Unfolding problem (Sun et al., 2006), with the intention to solve it via distributed globally optimal algorithms. In particular, we consider primal-dual subgradient iterations based on the dual of the original problem. We prove the convergence of these iterations to a global optimizer of the original non-convex problem under standard assumptions, even when approximations are involved in the distributed determination of the subgradients.

Furthermore, we illustrate a known relationship of the dual of the non-convex Maximum Variance Unfolding problem with the Fastest Mixing Markov Process problem (Sun et al., 2006), which is important in sensor network applications. This link allows us to use the proposed primal-dual scheme to solve both problems at the same time.

Finally, we discuss via a numerical simulation the performance of the proposed algorithm in a sensor network example and we show its relevance to more complex localization problems. Although this example involves fixed nodes, we argue in Section 5.5 how it could be extended to mobile nodes (robots). In addition, we illustrate that this mobile scenario is a generalization of the dispersion problem in robotic networks (Dimarogonas and Kyriakopoulos, 2009), where a group of robots strive to maximize the distance among themselves maintaining a certain specified connectivity of their underlying communication graph. This observation could offer ways to solve the dispersion problem in a distributed fashion.

5.2 Convex Networked Optimization under Resource Allocation Constraints

5.2.1 Problem Formulation

In this section, we study constrained convex optimization problems on a network of computing nodes. The network is modeled as a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices (nodes) in the set $\mathcal{V} = \{1, \dots, N\}$ and pairs of nodes as edges in the set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Consistent with the other chapters, we denote the cardinality of $\mathcal{E}$ as E, the set of neighbors of node i as $\mathcal{N}_i = \{j|(i,j) \in \mathcal{E}\}$, while L is the Laplacian of the graph $\mathcal{G}$.

We consider the following convex constrained optimization problem

$$\begin{aligned}
\underset{x_1,\dots,x_N}{\textbf{minimize}} \quad & \textstyle\sum_{i=1}^N f_i(x_i) && && (5.1a)\\
\textbf{subject to} \quad & g_{ij}(x_i, x_j) \le 0 && \text{for all } (i,j) \in \mathcal{E} && (5.1b)\\
& h_i(x_i) \le 0, && \text{for } i = 1, \dots, N && (5.1c)\\
& \textstyle\sum_{i=1}^N x_i = x_{\text{tot}} && && (5.1d)
\end{aligned}$$

where each variable $x_i \in \mathbb{R}^n$ is associated with the node i, all the functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $g_{ij} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable convex functions. The constant vector $x_{\text{tot}} \in \mathbb{R}^n$ dictates the total amount of each resource that is available in the system and therefore defines the resource allocation constraint[1]. We refer to problem (5.1) as the primal problem.

Let $x \in \mathbb{R}^{nN}$ be the stacked vector $x = (x_1^\top, \dots, x_N^\top)^\top$, while $f(x)$ is a separable cost function $f(x) = \sum_{i=1}^N f_i(x_i)$, and $g(x)$ denotes in a compact stacked form all the separable constraint functions described in (5.1b)-(5.1c), with codomain (or target set) $\mathbb{R}^m$, $m = E + N$. With this notation we can rewrite problem (5.1) in the compact form

$$\begin{aligned}
\underset{x}{\textbf{minimize}} \quad & f(x) && (5.2a)\\
\textbf{subject to} \quad & g(x) \le 0 && (5.2b)\\
& \left(\mathbf{1}_N^\top \otimes I_n\right) x = x_{\text{tot}} && (5.2c)
\end{aligned}$$

[1]For simplicity we consider here the case of $x_{\text{tot}} \in \mathbb{R}^n$, although our analysis could be extended to handle cases in which $x_{\text{tot}} \in \mathbb{R}^{\bar{n}}$, with $\bar{n} < n$. In this case the constraint (5.1d) need to be substituted with $\sum_{i=1}^N V x_i = x_{\text{tot}}$, where V is a matrix that selects the used $\bar{n}$ component.

We assume that the constraints $g(x) \le 0$ define a closed and bounded convex set $\bar{\mathbb{X}}$. We assume also that the intersection of $\bar{\mathbb{X}}$ and the set defined by the equality constraint $\left(\mathbf{1}_N^\top \otimes I_n\right) x = x_{\text{tot}}$ is a closed, bounded, and non-empty convex set, denoted by $\mathbb{X}$. Under these assumptions there exists a (possibly non-unique) optimizer of (5.1), which we indicate as x^{opt}, while we denote the primal optimal value by f^{opt}.

Problems of type (5.1) can be found in various domains including economics (Arrow and Hurwicz, 1960) and sensor networks (Palomar and Chiang, 2006). Typically, in these fields the nodes have a coupling resource allocation constraint, for example the total monetary budget or the total available bandwidth, respectively.

We are interested in solving the primal problem (5.1) via the use of iterative distributed algorithms. However, due to the facts that *(i)* the Lagrangian function associated with problem (5.1) is in general neither strictly convex in the primal variable x, nor strictly concave in the dual variables, and *(ii)* the resource allocation constraint couples all the nodes, standard primal-dual iterative methods have typically slow convergence rate and require high communication demand among the nodes. In order to address these issues, we study a regularized version of the saddle-point algorithm (primal-dual iterations) in the next section, which incorporates the resource allocation constraint directly in the update equations.

5.2.2 Regularized Saddle-Point Algorithm

In this section, we present a distributed gradient-based optimization method that employs a fixed regularization in the primal and dual spaces. This regularization serves to approximate the primal problem (5.1) in a way that can be solved by gradient-based methods with improved convergence properties. Furthermore, we modify the primal iteration to ensure that each iterate satisfies the resource allocation constraint. This allows us to avoid the dualization of the equality constraint, which would need to be distributed among the nodes and lead to increased communication requirements.

Let $\mu \in \mathbb{R}_+^m$ be the dual variable associated with the inequality constraint $g(x) \le 0$, and $\nu > 0$, $\epsilon > 0$ be strictly positive scalars. Motivated by (Koshal et al., 2011), we define a regularized Lagrangian-type function associated to the primal problem (5.1) as

$$\mathcal{L}(x,\mu) := f(x) + \frac{1}{2}\nu||x||^2 + \mu^\top g(x) - \frac{1}{2}\epsilon||\mu||^2 \tag{5.3}$$

This Lagrangian-type function is by definition a strictly convex function of the primal variable x and a strictly concave function of the dual variable μ.

In order to leverage on strong duality relations, we use the following standard assumption.

Assumption 5.1 *There exists a Slater vector $\bar{x} \in \mathbb{X}$ such that $g(\bar{x}) < 0$.*

Our aim is to find an (approximate) solution of the primal problem (5.1), by solving the regularized saddle-point problem:

$$\begin{aligned} \max_{\mu} \min_{x} \quad & \mathcal{L}(x,\mu) \\ \text{subject to} \quad & \left(\mathbf{1}_N^\top \otimes I_n\right) x = x_{\text{tot}} \end{aligned} \tag{5.4}$$

whose optimal value is denoted by f^*, and unique optimizer by x^*. Under Assumption 5.1, the unique optimizer of the regularized problem (5.4) satisfies the KKT conditions:

$$\nabla_x \mathcal{L}(x^*,\mu^*) + \left(\mathbf{1}_N^\top \otimes I_n\right)^\top p^* = 0 \tag{5.5a}$$

$$\nabla_\mu \mathcal{L}(x^*,\mu^*) \leq 0 \tag{5.5b}$$

$$\left(\mathbf{1}_N^\top \otimes I_n\right) x - x_{\text{tot}} = 0 \tag{5.5c}$$

$$\mu_q^* \nabla_{\mu_q} \mathcal{L}(x^*,\mu^*) = 0 \quad \text{for } q = 1,\dots,M \tag{5.5d}$$

$$\mu_q^* \geq 0 \quad \text{for } q = 1,\dots,M \tag{5.5e}$$

where μ^* and p^* are the optimal Lagrangian multipliers[2], while $\nabla_x\mathcal{L}$ and $\nabla_\mu\mathcal{L}$ indicate the gradients of the regularized Lagrangian-type function $\mathcal{L}(x,\mu)$ with respect to x and μ.

It is expected that in general the solutions of the primal problem (5.1) and the regularized saddle-point problem (5.4) are different, meaning $||x^* - x^{\text{opt}}|| \neq 0$ and $||f^* - f^{\text{opt}}|| \neq 0$. Furthermore, the solution of the regularized problem (5.4) does not necessarily satisfy the inequality constraints of the primal problem (5.1). However, it is possible to bound the suboptimality and the distance from the primal optimizer, along with the constraint violation by some function of the regularization parameters ν and ϵ. Thus while we are solving an approximation of the primal problem (5.1) we have bounds on the distance from the primal optimal solution. Furthermore, in this context the regularization procedure can be seen as a way to speed up the convergence of standard gradient-like methods, which may in fact lead to a closer iterate to the optimum f^{opt} of the primal problem within a finite number of iterations even though an approximate regularized problem is being solved. For further details we refer the reader to the original works on regularization and double smoothing techniques (Devolder et al., 2011, Koshal et al., 2011).

The regularized saddle-point problem (5.4) can be readily solved by centralized iterative methods. However, when a distributed solution is sought, the equality constraint is usually dualized and decomposed among the nodes, see for example the discussions in (Jadbabaie et al., 2009, Zhu and Martínez, 2012). Typically, this procedure causes high communication load and the convergence rate would be affected by the number of nodes. In order to overcome these potential drawbacks we follow a different route and propose a method that ensures the feasibility of each iterate with respect to the resource allocation constraint. The main idea can be thought of as "projecting" the iterates onto the feasible set of the equality constraint. This extension allows us to design an inherently distributed iterative scheme that still solves the original regularized problem (5.4).

Let $\mathcal{P}_{\mathbb{R}_+}$ indicate the projection over the positive orthant, and let $\alpha > 0$ and $\beta > 0$ be fixed

[2]The Lagrangian multiplier $p \in \mathbb{R}^n$ corresponds to the resource allocation equality constraint, which is used to describe the KKT conditions of the regularized problem, but not used in our proposed iterative solution approach.

strictly positive scalars (step sizes). We consider the following saddle-point iterations:

$$x^{(\tau+1)} = x^{(\tau)} - \alpha\beta(W \otimes I_n)\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) \tag{5.6}$$

$$\mu^{(\tau+1)} = \mathcal{P}_{\mathbb{R}_+}\left[\mu^{(\tau)} + \alpha\nabla_\mu \mathcal{L}(x^{(\tau)}, \mu^{(\tau)})\right] \tag{5.7}$$

with any matrix $W \in \mathbb{R}^{N\times N}$ such that

(a) the vectors $\mathbf{1}_N$ and $\mathbf{1}_N^\top$ are left and right eigenvectors of W associated to the zero eigenvalue, respectively:

$$\mathbf{1}_N^\top W = \mathbf{0}, \qquad W\mathbf{1}_N = \mathbf{0}$$

(b) the zero eigenvalue is unique, i.e.,

$$W + W^\top + (1/N)\mathbf{1}_N\mathbf{1}_N^\top \succ 0$$

(c) the matrix W has the same sparsity pattern as the Laplacian matrix L of the graph $\mathcal{G}$.

It is easy to see that if the properties *(a)*-*(c)* hold, then the iterations (5.6)-(5.7) can be computed locally with only the information of the neighboring nodes. In this sense, the iterations (5.6)-(5.7) are inherently distributed.[3]

[3]In order to explicitly see this, let q_{ij} be the Lagrangian multiplier associated with $g_{ij}(x_i, x_j) \le 0$ and let v_i be the one associated with $h_i(x_i) \le 0$ in problem (5.1) (we recall that $\mu = (q^\top, v^\top)^\top$). The Lagrangian-type function in (5.3) can be rewritten as

$$\mathcal{L}(x,q,v) = \sum_{i=1}^N f_i(x_i) + \frac{1}{2}\nu\sum_{i=1}^N ||x_i||^2 + \sum_{(i,j)\in\mathcal{E}} q_{ij}g_{ij}(x_i,x_j) + \sum_{i=1}^N v_i h_i(x_i) - \frac{1}{2}\epsilon\left(\sum_{(i,j)\in\mathcal{E}} ||q_{ij}||^2 + \sum_{i=1}^N ||v_i||^2\right)$$

and therefore the update rule (5.6) can be written as

$$\begin{aligned} x_i^{(\tau+1)} &= x_i^{(\tau)} - \alpha\beta(W\otimes I_n)\cdot \\ &\quad \begin{pmatrix} \nabla_{x_1} f_1(x_1^{(\tau)}) + 2\nu x_1^{(\tau)} + \sum_{j\in\mathcal{N}_1} q_{1j}^{(\tau)}\nabla_{x_1} g_{1j}(x_1^{(\tau)}, x_j^{(\tau)}) + v_1^{(\tau)}\nabla_{x_1} h_1(x_1^{(\tau)}) \\ \vdots \\ \nabla_{x_N} f_N(x_N^{(\tau)}) + 2\nu x_N^{(\tau)} + \sum_{j\in\mathcal{N}_N} q_{Nj}^{(\tau)}\nabla_{x_N} g_{Nj}(x_N^{(\tau)}, x_j^{(\tau)}) + v_N^{(\tau)}\nabla_{x_N} h_N(x_N^{(\tau)}) \end{pmatrix} \\ &= x_i^{(\tau)} - \alpha\beta\sum_{j\in\mathcal{N}_i^+} W_{ij}\Bigg(\nabla_{x_j} f_j(x_j^{(\tau)}) + 2\nu x_j^{(\tau)} + \sum_{\ell\in\mathcal{N}_j} q_{j\ell}^{(\tau)}\nabla_{x_j} g_{j\ell}(x_j^{(\tau)}, x_\ell^{(\tau)}) + \\ &\qquad v_j^{(\tau)}\nabla_{x_j} h_j(x_j^{(\tau)})\Bigg) \end{aligned}$$

while the update (5.7) as $q_{ij}^{(\tau+1)} = \mathcal{P}_{\mathbb{R}_+}\left[q_{ij}^{(\tau)} + \alpha\left(\nabla_{q_{ij}} g_{ij}(x_i^{(\tau)}, x_j^{(\tau)}) - \epsilon q_{ij}^{(\tau)}\right)\right]$ and $v_i^{(\tau+1)} = \mathcal{P}_{\mathbb{R}_+}\left[v_i^{(\tau)} + \alpha\left(\nabla_{v_i} h_i(x_i^{(\tau)}) - \epsilon v_i^{(\tau)}\right)\right]$.

We claim that there exist conditions on the step sizes α and β such that the iterations (5.6)-(5.7) converge to the unique optimal solution of the regularized problem (5.4). In particular, we expect that the step size α is linked to the characteristics of the functions f and g (as in standard gradient-like methods), while β is linked to W, i.e., the network topology. These relationships will be shown using the following lemma, which establishes three important properties of the iterations (5.6)-(5.7).

Lemma 5.1 *If the matrix W satisfies the property* (a) *and for the first iterate $x^{(0)}$ the resource allocation constraint holds, i.e., $(\mathbf{1}_N \otimes I_n)^\top x^{(0)} = x_{\text{tot}}$, then*

(i) *for any τ, the iterate $x^{(\tau)}$ satisfies the resource allocation constraint, i.e., $(\mathbf{1}_N \otimes I_n)^\top x^{(\tau)} = x_{\text{tot}}$;*

(ii) *the optimal couple (x^*, μ^*) of* (5.4) *is a fixed point of the iterations* (5.6)-(5.7)*;*

(iii) *for any τ, the equality $x^{(\tau+1)} = x^{(\tau)}$ holds if and only if either $\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) + (\mathbf{1}_N \otimes I_n)p = 0$, for some $p \in \mathbb{R}^n$, or $\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) = \mathbf{0}$.*

Proof. The first claim follows by induction based on (Xiao and Boyd, 2006). Suppose that $x^{(\tau)}$ satisfies the resource allocation constraint. Then for $x^{(\tau+1)}$

$$\left(\mathbf{1}_N^\top \otimes I_n\right) x^{(\tau+1)} = \left(\mathbf{1}_N^\top \otimes I_n\right) \left(x^{(\tau)} - \alpha\beta(W \otimes I_n)\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)})\right)$$

and using the fact that $\left(\mathbf{1}_N^\top \otimes I_n\right)(W \otimes I_n) = \mathbf{1}_N^\top W \otimes I_n = \mathbf{0}$ (property *(a)* of W) the claim follows.

The second claim follows by direct calculations. Consider the optimal pair (x^*, μ^*) of (5.4), then using the KKT conditions we obtain

$$\begin{aligned} x^{(\tau+1)} = x^* - \alpha\beta(W \otimes I_n)\nabla_x \mathcal{L}(x^*, \mu^*) &= x^* + \alpha\beta\left(W \otimes I_n\right)\left(\mathbf{1}_N^\top \otimes I_n\right)^\top p^* \\ &= x^* + \alpha\beta\left(W \otimes I_n\right)\left(\mathbf{1}_N \otimes I_n\right)p^*. \end{aligned}$$

Since $(W \otimes I_n)(\mathbf{1}_N \otimes I_n) = W\mathbf{1}_N \otimes I_n = \mathbf{0}$, it follows that $x^{(\tau+1)} = x^*$ and therefore x^* is a fixed point.

The third claim follows from property *(b)* of W, i.e., the uniqueness of the zero eigenvalue. The equality $x^{(\tau+1)} = x^{(\tau)}$ holds if and only if $\alpha\beta(W \otimes I_n)\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) = \mathbf{0}$. This last equality is true either if $\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) = \mathbf{0}$ or if the vector $\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)})$ is an eigenvector of W with associated zero eigenvalue. Therefore, using property *(b)* of W leads to $\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) = (\mathbf{1}_N \otimes I_n)p'$, with $p' \in \mathbb{R}^n$. Choosing $p' = -p$ proves the claim. □

Lemma 5.1 shows that the information exchange matrix W keeps the iterates feasible with respect to the resource allocation constraint and does not introduce undesired fixed points.

The next section investigates the conditions on α and β under which the primal-dual iterates $x^{(\tau)}$ and $\mu^{(\tau)}$ converge to the optimizer (x^*, μ^*) of (5.4), and the bounds on how far this solution is from the primal solution x^{opt} in terms of suboptimality $||f^* - f^{\text{opt}}||$ and constraint violation.

Furthermore, we will show that the choice of the information exchange matrix W (especially its largest eigenvalue) is important for convergence and we will mention how one could improve the performances of the iterates tuning the weights in W.

5.2.3 Convergence Properties

Let z be the stacked vector $z = (x^\top, \mu^\top)^\top$, and define the mapping

$$\Phi(z) = (\nabla_x \mathcal{L}(z)^\top, -\nabla_\mu \mathcal{L}(z)^\top)^\top.$$

We use the short-hand notation $W_\otimes = W \otimes I_n$. Moreover, let $\mathcal{P}$ be a generic projection operator, whose codomain will be clear by the context. The iterations (5.6)-(5.7) can be compactly written as:

$$z^{(\tau+1)} = \mathcal{P}\left[z^{(\tau)} - \alpha \begin{bmatrix} \beta W_\otimes & \\ & I_M \end{bmatrix} \Phi(z^{(\tau)})\right] =: T(z^{(\tau)}) \tag{5.8}$$

The scope of this section is to identify the assumptions on $\mathcal{L}(x, \mu)$ and the conditions on α and β that let the mapping $T : \mathbb{R}^{Nn+m} \to \mathbb{R}^{Nn+m}$ be a contraction mapping. This guarantees geometric convergence of the iterations (5.6)-(5.7) to the optimal point of (5.4).

First of all we characterize the properties of the mapping $\Phi(z)$ under the following assumptions.

Assumption 5.2 *The iterates $x^{(\tau)}$ and $\mu^{(\tau)}$ are contained in some closed, convex, and bounded sets for each iteration τ. In other words, $x^{(\tau)} \in \hat{\mathbb{X}}$ and $\mu^{(\tau)} \in \hat{\mathbb{M}}$, with $\hat{\mathbb{X}}$ and $\hat{\mathbb{M}}$ closed, convex, and bounded sets.*

We note that the assumption of $\mu^{(\tau)} \in \hat{\mathbb{M}}$ is satisfied under Assumption 5.1 (see Koshal et al. (2011) for details).

On the other hand, the assumption on $x^{(\tau)}$ is a stronger requirement. We remark that we cannot remove this assumption by enforcing it in the iterations on $x^{(\tau)}$, i.e., projecting $x^{(\tau+1)}$ onto some closed, convex, and bounded set, since this would destroy the properties of the information exchange matrix W. However, one could think of enforcing it via a modification of the Lagrangian-type function (5.3) through a barrier function. A formal characterization of this modification is currently under investigation.[4]

We also make the following mild and technical assumptions:

Assumption 5.3 *The gradients of $f(x)$ and each $g_q(x)$ are Lipschitz continuous with constants F and G_q, respectively:*

$$\begin{aligned} ||\nabla_x f(a) - \nabla_x f(b)|| &\le F||a-b||, &&\textit{for all } a, b \in \hat{\mathbb{X}} \\ ||\nabla_x g_q(a) - \nabla_x g_q(b)|| &\le G_q||a-b||, &&\textit{for all } a, b \in \hat{\mathbb{X}}, q = 1, \ldots, m \end{aligned}$$

[4] We note that we could also pick $\hat{\mathbb{X}}$ as $\mathbb{R}^{Nn}$, if f and g satisfied Assumptions 5.3-5.4 with this choice.

Assumption 5.4 *The constraint gradient and Lagrangian dual variable are bounded as*

$$||\nabla_x g(x)|| < M_d, \text{ and } ||\mu|| \leq M_\mu.$$

We note that Assumptions 5.3-5.4 are commonly required in the analysis of gradient descent methods. Furthermore, Assumption 5.4 is generally satisfied under Assumption 5.2.

Assumptions 5.1, 5.3, and 5.4 are important to guarantee that the mapping $\Phi(z)$ has certain regularity properties. In fact, under these assumptions, by Lemma 3.4 of (Koshal et al., 2011), the mapping $\Phi(z)$ is strongly monotone with constant $\varphi = \min(\nu, \epsilon)$ and Lipschitz with constant F_Φ. In other words,

$$\langle \Phi(a) - \Phi(b), a - b \rangle \geq \varphi ||a - b||^2, \text{ for } a, b \in \hat{\mathbb{X}} \times \hat{\mathbb{M}} \tag{5.9}$$

$$||\Phi(a) - \Phi(b)|| \leq F_\Phi ||a - b||, \text{ for } a, b \in \hat{\mathbb{X}} \times \hat{\mathbb{M}} \tag{5.10}$$

Properties (5.9) and (5.10) will be important for convergence.

Symmetric Case

In this subsection we will assume that the matrix W is symmetric, i.e., $W^\top = W$. This will allow us to derive closed-form conditions for the step-sizes α and β. Define

$$C := \max(\beta \lambda_{\max}(W), 1) \tag{5.11}$$

$$\kappa := \varphi - F_\Phi \left(\beta \lambda_{\max}(W) - 1\right) \tag{5.12}$$

where $\lambda_{\max}(W)$ is the maximum eigenvalue of W (which can be upper-bounded in a distributed way, e.g. (Li and Pan, 2001)). Define the ratio

$$r = \frac{||z^{(\tau+1)} - z^*||^2}{||z^{(\tau)} - z^*||^2}.$$

If $r < 1$ we say that the sequence $\{z^{(\tau)}\}$ has geometrical convergence to $z^\star$ and its convergence rate is r.

Theorem 5.1 *Under the Assumptions 5.1, 5.2, 5.3, and 5.4 and for symmetric W, the conditions*

$$\beta \lambda_{\max}(W) < 1 + \frac{\varphi}{F_\Phi}, \quad \text{and} \quad \alpha < \frac{2\kappa}{C^2 F_\Phi^2} \tag{5.13}$$

ensure geometrical convergence of the iterations (5.6)-(5.7) to the unique optimizer (x^*, μ^*) of the regularized problem (5.4). Furthermore, the convergence rate r is

$$r = 1 - 2\alpha\kappa + \alpha^2 C^2 F_\Phi^2 \tag{5.14}$$

Proof. The distance of the primal iterate $x^{(\tau+1)}$ to a primal optimizer x^* can be written as

$$||x^{(\tau+1)} - x^*||^2 = ||x^{(\tau)} - x^*||^2 -$$

$$2\alpha \left\langle \beta W_{\otimes} \left(\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right), x^{(\tau)} - x^* \right\rangle + \\ \alpha^2 \left\| \beta W_{\otimes} \left(\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right) \right\|^2 \quad (5.15)$$

where we made use of the fact that x^* is a fixed point of the iteration (5.6). Using the relation

$$\left\| \beta W_{\otimes} \left(\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right) \right\|^2 \leq \\ \beta^2 \lambda^2_{\max}(W) \left\| \nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right\|^2$$

equation (5.15) becomes

$$||x^{(\tau+1)} - x^*||^2 \leq ||x^{(\tau)} - x^*||^2 - \\ 2\alpha \left\langle \beta W_{\otimes} \left(\nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right), x^{(\tau)} - x^* \right\rangle + \\ \alpha^2 \beta^2 \lambda^2_{\max}(W) \left\| \nabla_x \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_x \mathcal{L}(x^*, \mu^*) \right\|^2 \quad (5.16)$$

In a similar fashion, and using the non-expansive property of the projection, we can write the distance of the Lagrangian multiplier $\mu^{(\tau+1)}$ to its optimal value μ^* as:

$$||\mu^{(\tau+1)} - \mu^*||^2 \leq ||\mu^{(\tau)} - \mu^*||^2 + \\ 2\alpha \left\langle \nabla_\mu \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_\mu \mathcal{L}(x^*, \mu^*), \mu^{(\tau)} - \mu^* \right\rangle + \\ \alpha^2 \left\| \nabla_\mu \mathcal{L}(x^{(\tau)}, \mu^{(\tau)}) - \nabla_\mu \mathcal{L}(x^*, \mu^*) \right\|^2 \quad (5.17)$$

Summing up the relations (5.16)-(5.17) we obtain

$$||z^{(\tau+1)} - z^*||^2 \leq ||z^{(\tau)} - z^*||^2 - \\ 2\alpha \left\langle \begin{bmatrix} \beta W_{\otimes} & \\ & I_M \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*) \right), z^{(\tau)} - z^* \right\rangle + \\ \alpha^2 C^2 \left\| \Phi(z^{(\tau)}) - \Phi(z^*)) \right\|^2 \quad (5.18)$$

where C is defined as in (5.11). The term

$$- \left\langle \begin{bmatrix} \beta W_{\otimes} & \\ & I_M \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*) \right), z^{(\tau)} - z^* \right\rangle$$

can be expanded as

$$- \left\langle \begin{bmatrix} \beta W_{\otimes} & \\ & I_M \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*) \right), z^{(\tau)} - z^* \right\rangle =$$

$$\underbrace{-\left\langle \begin{bmatrix} I_{Nn} & \\ & I_M \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*)\right), z^{(\tau)} - z^* \right\rangle}_{\text{(a)}} +$$
$$\underbrace{-\left\langle \begin{bmatrix} \beta W_\otimes - I_{Nn} & \\ & 0 \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*)\right), z^{(\tau)} - z^* \right\rangle}_{\text{(b)}}$$

We can bound the term (a) based on the strong monotonicity of $\Phi(z)$ in (5.9), while the term (b) can be bounded as

$$\begin{aligned} -\left\langle \begin{bmatrix} \beta W_\otimes - I_{Nn} & \\ & 0 \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*)\right), z^{(\tau)} - z^* \right\rangle \\ \leq \left\| \left\langle \begin{bmatrix} \beta W_\otimes - I_{Nn} & \\ & 0 \end{bmatrix} \left(\Phi(z^{(\tau)}) - \Phi(z^*)\right), z^{(\tau)} - z^* \right\rangle \right\| \\ \leq \left\| \begin{bmatrix} \beta W_\otimes - I_{Nn} & \\ & 0 \end{bmatrix} \right\| \left\| \Phi(z^{(\tau)}) - \Phi(z^*) \right\| \left\| z^{(\tau)} - z^* \right\| \\ \leq \left(\beta \lambda_{\max}(W) - 1\right) F_\Phi ||z^{(\tau)} - z^*||^2 \end{aligned}$$

where we used the Lipschitz continuity of $\Phi(z)$ in (5.10). The relation (5.18) then becomes

$$||z^{(\tau+1)} - z^*||^2 \leq \left(1 - 2\alpha\kappa + \alpha^2 C^2 F_\Phi^2\right) ||z^{(\tau)} - z^*||^2 \tag{5.19}$$

with κ defined as in (5.12). Therefore the first convergence condition is

$$1 - 2\alpha\kappa + \alpha^2 C^2 F_\Phi^2 < 1,$$

while, since it is required that $\alpha > 0$, the second condition must be $\kappa > 0$. From these two conditions the relations (5.13) follow. Furthermore the convergence rate expression in (5.14) can be established based on (5.19). □

Corollary 5.1 *The convergence conditions* (5.13) *on the step sizes are satisfied if the following, more conservative, conditions are met*

$$\beta < \frac{1}{\lambda_{\max}(W)}, \quad \textit{and} \quad \alpha < 2\frac{\varphi}{F_\Phi^2} \tag{5.20}$$

Proof. The proof follows directly from $\varphi / F_\Phi > 0$. □

The type of conditions in Corollary 5.1 are typical in (sub)gradient methods and are often referred to as “small enough” step size conditions (Bertsekas, 1999). We may notice that α is bounded by quantities related to the characteristics of the problem functions, while β is related to the structure of the information exchange graph. We also note that α has to be determined a priori based on the knowledge of the problem function properties, while β can be computed in a distributed way by the nodes, since there are distributed algorithms to upper-bound $\lambda_{\max}(W)$, e.g. (Li and Pan, 2001).

The following technical lemma characterizes the “quality” of the regularized optimal solution x^* with respect to the original primal problem (5.1) in function of the regularization parameters ν and ϵ: it provides bounds on the amount of constraint violation of $g(x^*)$ and the suboptimality $||f^* - f^{\text{opt}}||$. In order to compactly characterize these bounds, we define the constraint set of the regularized problem (5.4) as $\mathbb{X}_{\nu,\epsilon}$, which implies that $x^* \in \mathbb{X}_{\nu,\epsilon}$. The set $\mathbb{X}_{\nu,\epsilon}$ is closed, bounded, and convex, and in general different from the original primal constraint set $\mathbb{X}$, being however $\mathbb{X} \subseteq \mathbb{X}_{\nu,\epsilon}$.

Lemma 5.2 *Under the Assumptions 5.1, 5.2, 5.3, and 5.4, and using ν, ϵ as regularization parameters in (5.3) the maximum constraint violation is bounded by*

$$\max\{0, g_i(x^*)\} \leq M_{d_i} M_\mu \sqrt{\frac{\epsilon}{2\nu}} \tag{5.21}$$

where $M_{d_i} = \max_{x\in\hat{\mathbb{X}}} ||\nabla_x g_i(x)||$ for each i and $M_\mu = \max_{\mu\in\hat{\mathbb{M}}} ||\mu||$, while the difference between the optimal value of the regularized problem (5.4) and the optimal value of the original one (5.1) can be bounded by

$$||f^* - f^{\text{opt}}|| \leq M_f M_\mu \sqrt{\frac{\epsilon}{2\nu}} + \frac{\nu}{2} D^2 \tag{5.22}$$

where $M_f = \max_{x\in\mathbb{X}_{\nu,\epsilon}} ||\nabla_x f(x)||$, $D = \max_{x\in\mathbb{X}_{\nu,\epsilon}} ||x||$

Proof. The proof is a modified version of Lemma 3.3 in (Koshal et al., 2011). In particular, the bound (5.21) follows directly from Lemma 3.3 in (Koshal et al., 2011), while the bound (5.22) requires the modification that we consider for M_f and D the maximum of x over $\mathbb{X}_{\nu,\epsilon}$ instead of $\hat{\mathbb{X}}$ (which could lead to a too conservative result in our case).

The proof of the bound (5.22) starts from bounding $||f^* - f^{\text{opt}}||$ by

$$||f^* - f^{\text{opt}}|| \leq ||f^* - f^*_{\epsilon=0}|| + f^*_{\epsilon=0} - f^{\text{opt}} \tag{5.23}$$

where $f^*_{\epsilon=0}$ is the optimal cost for the regularization problem with regularization parameter $\epsilon = 0$, and $f^*_{\epsilon=0} - f^{\text{opt}} \geq 0$. By convexity of f, we have

$$f^* - f^*_{\epsilon=0} \leq \nabla_x f(x^*)^\top (x^* - x^*_{\epsilon=0}) \tag{5.24}$$

with $x^*_{\epsilon=0}$ the unique optimizer of the regularization problem with regularization parameter $\epsilon = 0$. Since $x^*, x^*_{\epsilon=0} \in \mathbb{X}_{\nu,\epsilon}$ and $\mathbb{X}_{\nu,\epsilon}$ is compact, by the continuity of the gradient $||\nabla_x f(x)||$, the gradient norm is bounded and we can write (5.24) as

$$||f^* - f^*_{\epsilon=0}|| \leq \underbrace{\max_{x\in\mathbb{X}_{\nu,\epsilon}} ||\nabla_x f(x)||}_{M_f} ||x^* - x^*_{\epsilon=0}|| \tag{5.25}$$

However, By Proposition 3.1 of (Koshal et al., 2011), we can bound $||x^* - x^*_{\epsilon=0}||$ by $M_\mu \sqrt{\epsilon/2\nu}$ and therefore (5.25) can be written as

$$||f^* - f^*_{\epsilon=0}|| \leq M_f M_\mu \sqrt{\frac{\epsilon}{2\nu}} \tag{5.26}$$

Using the estimate $f^*_{\epsilon=0} - f^{\text{opt}} \leq \nu/2 \max_{x\in\mathbb{X}_{\nu,\epsilon}} ||x||^2$, which follows directly from the definition of the cost function (see Lemma 7.1 of (Koshal et al., 2011)), then (5.23) can be written as

$$||f^* - f^{\text{opt}}|| \leq M_f M_\mu \sqrt{\frac{\epsilon}{2\nu}} + \frac{\nu}{2} D^2$$

□

Non-symmetric Case

In this subsection, we consider the case of non-symmetric matrices W. It is not difficult to see that all the previous derivations hold true with the small modification that instead of $\lambda_{\max}(\cdot)$ we will have $\sigma_{\max}(\cdot)$, meaning the largest singular value. Unfortunately, due to the term $\sigma_{\max}(\beta W - I_N)$, the condition on β is not solvable in closed-form. In particular, if we define

$$C' := \max(\beta\sigma_{\max}(W), 1), \quad \kappa' := \varphi - F_\Phi \sigma_{\max}(\beta W - I_N)$$

then the conditions in (5.13) for the non-symmetric case become

$$\sigma_{\max}(\beta W - I_N) < \frac{\varphi}{F_\Phi}, \quad \text{and} \quad \alpha < \frac{2\kappa'}{C'^2 F_\Phi^2}.$$

Weight Design

Instead of a unique step size β, one may consider designing the whole information exchange weight matrix W. For simplicity we redefine the weight matrix as $\overline{W} := \beta W$ whose pattern is fixed by the network structure (and supposed to be symmetric) but the single entries are variables to be determined. If we use $\overline{W}$ in the iterations (5.6)-(5.7), the convergence conditions on $\overline{W}$ (in addition to the one on α) can be written as

$$\begin{aligned} \lambda_2(\overline{W}) > 0 \quad &\Longleftrightarrow \quad \overline{W} + (1/N)\mathbf{1}_N\mathbf{1}_N^\top \succ 0 \\ \lambda_{\max}(\overline{W}) < 1 \quad &\Longleftrightarrow \quad \overline{W} - I_N \prec 0 \end{aligned}$$

These conditions are similar to those in (Xiao and Boyd, 2006). In particular, the first condition is a connectivity condition, while the second could be interpreted as diagonal dominance. Using the fact that $(\overline{W} + (1/N)\mathbf{1}_N\mathbf{1}_N^\top)^{-1}\overline{W} = (I_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)$ and therefore

$$\overline{W}(\overline{W} + (1/N)\mathbf{1}_N\mathbf{1}_N^\top)^{-1}\overline{W} = \overline{W}$$

by Schur complement these relations can be translated into the LMI (Xiao and Boyd, 2006)

$$\begin{bmatrix} \overline{W} + (1/N)\mathbf{1}_N\mathbf{1}_N^\top & \overline{W} \\ \overline{W} & I_N \end{bmatrix} \succ 0 \tag{5.27}$$

which makes the weight design a centralized convex problem. Finally, one could even optimize the weights to improve performance; this could be done by the centralized convex problem

$$\begin{aligned} \underset{\gamma\geq 0, W}{\textbf{maximize}} \quad & \gamma && (5.28a)\\ \textbf{subject to} \quad & \overline{W} \succeq 0, \quad \overline{W}\mathbf{1}_N = \mathbf{0} && (5.28b)\\ & \begin{bmatrix} \overline{W} + (1/N)\mathbf{1}_N\mathbf{1}_N^\top & \overline{W} \\ \overline{W} & I_N \end{bmatrix} \succ \gamma I_{2N}. && (5.28c) \end{aligned}$$

5.2.4 Summary of the Proposed Solution

We summarize the proposed regularized saddle-point iterations in Algorithm 5.1. We recall once more that:

- The choice of the regularization parameters ν and ϵ determines the convergence properties of Algorithm 5.1. The higher the parameters are, the faster the convergence rate could be. However, the higher these parameters are, the further away is the approximate solution of Algorithm 5.1 to the real optimum of the optimization problem (5.1).
- The choice of the step sizes α and β is determined via the convergence results Theorem 5.1 and Corollary 5.1. In particular, the step size β has to be determined by the knowledge of the maximum eigenvalue of the network Laplacian (which can be computed in a distributed way by the nodes, since there are distributed algorithms to upper-bound $\lambda_{\max}(W)$, e.g. (Li and Pan, 2001)). The step size α is instead linked to the properties of the optimization problem (not of the graph but of the optimization functions), and Theorem 5.1 ensures that there exists a "small enough" α that guarantees convergence.

Algorithm 5.1 Regularized Saddle-Point Algorithm

```
1: Input: $x^{(\tau)}, \mu^{(\tau)}$
   ▷ Available data: $f, g, \alpha, \beta, \nu, \epsilon, W$
2: Compute: $\nabla_x\mathcal{L}(x^{(\tau)}, \mu^{(\tau)})$ and $\nabla_\mu\mathcal{L}(x^{(\tau)}, \mu^{(\tau)})$ with
       $$\mathcal{L}(x,\mu) = f(x) + \frac{1}{2}\nu||x||^2 + \mu^\top g(x) - \frac{1}{2}||\mu||^2$$
3: Compute:
       $$x^{(\tau+1)} = x^{(\tau)} - \alpha\beta(W \otimes I_n)\nabla_x\mathcal{L}(x^{(\tau)}, \mu^{(\tau)})$$
       $$\mu^{(\tau+1)} = \mathcal{P}_{\mathbb{R}_+}\left[\mu^{(\tau)} + \alpha\nabla_\mu L(x^{(\tau)}, \mu^{(\tau)})\right]$$
4: Output: $x^{(\tau+1)}, \mu^{(\tau+1)}$
```

5.2.5 A Robotic Network Application: Target Tracking and Barycenter Keeping

In this section we use an application scenario inspired by a realistic problem to illustrate the proposed method. We consider a group of N mobile robots that can communicate among each other via a communication network. Let the graph that describes the network be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and we will assume that it is time-invariant. As usual, let $x_i(k) \in \mathbb{R}^2$ be the position of robot i at the discrete time step k. Let $y(k) \in \mathbb{R}^2$ be the position of a moving target at the discrete time step k. We model the robot dynamics as single integrator systems and associate the convex cost function $f_i(x_i(k) - x_i(k-1))$ with each of them that can represent the energy consumption. We assume the robots know the target location.

Motivated by recent research works on multi-target tracking and target circumnavigation (Derenick et al., 2009, Shames et al., 2012), we are interested in moving the robots to ensure that the target is always in the barycenter of their positions. Furthermore, we require that robots connected by an edge in the fixed graph have a bound R on their maximal distance (for communication purposes). We limit the allowable change of position in one step $||x_i(k) - x_i(k-1)||$ by $v_{\max,i}\Delta t$ to model physical limitations. Finally, our global objective is to meet the aforementioned requirements while minimizing the total energy consumption. At each discrete time k, the above problem can be written as

$$\begin{aligned} \underset{x_1(k),\dots,x_N(k)}{\textbf{minimize}} \quad & \textstyle\sum_{i=1}^{N} f_i(x_i(k) - x_i(k-1)) \\ \textbf{subject to} \quad & ||x_i(k) - x_j(k)||^2 - R^2 \le 0 \\ & \qquad\qquad \text{for all } (i,j) \in \mathcal{E} \\ & ||x_i(k) - x_i(k-1)|| - v_{\max,i}\Delta t \le 0 \\ & \qquad\qquad \text{for } i = 1,\dots,N \\ & 1/N \textstyle\sum_{i=1}^{N} x_i(k) = y(k) \end{aligned} \tag{5.29}$$

which is a specific instance of (5.1) for each time step k. In particular, since the target is moving, $y(k)$ corresponds to a time-varying total available resource x_{tot} in the formulation of problem (5.1)[5].

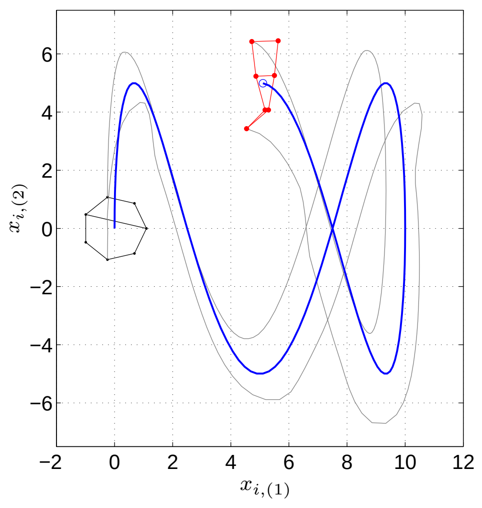
Our simulation example consists of $N = 7$ robots connected via a communication graph shown in Figure 5.1 with Laplacian matrix L. The parameters of the scenario are $R = 1.2$, and

$$f_i(\delta x_i(k)) = \langle Q_i \delta x_i(k), \delta x_i(k) \rangle,$$

where $\delta x_i(k) = x_i(k) - x_i(k-1)$ and $Q_i = 1$ for all i except for $i = 6$, for which $Q_6 = 0$. In practice, this translates in a non-strictly convex cost function. We consider $v_{\max,6} = 0.5$ (which means that the robot 6 is limited only by physical constraints, and not by the cost), while the other $v_{\max}$ are set to $+\infty$ (meaning that the other robots do not have physical constraints). Given the fact that the cost function is not strictly convex and the position of the robots are coupled via a resource allocation constraint, even this small-size problem could be difficult to solve (in terms of communication/computation requirements) for common gradient algorithms. This makes this example interesting to analyze with the proposed approach.

We solve problem (5.29) via the proposed regularized saddle-point Algorithm 5.1 with $\nu = 10$, $\epsilon = .01$, and $W = L$. The step sizes are $\beta = 0.2$ (determined via Corollary 5.1) and $\alpha = 0.01$ (determined via trial-and-error). In practice, since x_{tot} $(y(k))$ is varying we

[5] We note that, when the proposed saddle-point algorithm is used to solve the problem (5.29) at each discrete time step k, each initial $x_i(k)^{(0)}$ can be chosen as $x_i(k)^{(0)} = x_i(k-1) + (y(k) - y(k-1))$, with $x_i(0)^{(0)} = y(0)$. This ensures that the initial iterates satisfy the resource allocation constraint.



$$L=\begin{bmatrix} 3 & -1 & 0 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Figure 5.1: *Representation of the trajectories of two robots while the target moves (blue thick line). The initial graph and positions of the robots are marked in black, while the final configuration is marked in red. The notation $x_{i,(j)}$ indicates the j-th component of x_i.*

solve problem (5.29) at convergence for each k using Algorithm 5.1.

Figure 5.1 shows the computed trajectories of the robots while the target moves (blue thick line). The initial graph and positions of the robots are marked in black, while the final configuration is marked in red. In order to assess the "quality" of the regularized problem solution with respect to the original primal one, the maximal error of the optimal robot positions $\max_k ||x^*(k) - x^{\text{opt}}(k)||$ was computed and resulted in 0.02, which is acceptable in this application scenario. Finally, we report that the total number of communication/computation iterations per discrete time step k per robot was $\tau = 2000$, and the computations required around 0.03 s per node per discrete time step k, on an Intel Core *i5* (2.3 GHz and 4GB DDR3) laptop. These results are encouraging since the regularization parameters were not specifically optimized to minimize the number of iterations. This aspect, along with extensive comparisons with common gradient methods are left as future development. [6]

[6] As a preliminary result, we remark that dualizing the resource allocation constraint would cause an increase of the number of iterations of at least 40%, even with $\beta = 1$. We expect non-regularized gradient methods to need even more iterations to achieve the same accuracy as comprehensively illustrated in (Koshal et al., 2011).

5.3 The Maximum Variance Unfolding Problem

While the previous section we have looked into convex resource allocation problems, in this section we shift our focus on a particular non-convex one, known as Maximum Variance Unfolding problem, with the intention to solve it via global optimal distributed algorithms. This problem arises in different research areas, such as localization (Weinberger et al., 2007), mechanics (Sun et al., 2006), linear algebra (Göring et al., 2008), and unsupervised (machine) learning (Weinberger and Saul, 2006). Furthermore, together with its dual, it is linked to important research questions in robotic networks, such as the maximization and minimization of the algebraic connectivity of the underlying graph under given constraints.

5.3.1 Problem Formulation

With the notation of Section 5.2, let the optimization variable associated to node i be x_i and, for simplicity, consider this *nodal* variable to be scalar, i.e. $x_i \in \mathbb{R}$ (This assumption will be removed in Section 5.3.5). Let $r_{ij} \in \mathbb{R}$ be a bound associated with the edge (i,j) of the graph $\mathcal{G}$ connecting the nodes. We are interested in solving the following non-convex problem

$$\underset{x_1,\ldots,x_N}{\textbf{maximize}} \quad \sum_{i=1}^{N} ||x_i||^2 \tag{5.30a}$$

$$\textbf{subject to} \quad ||x_i - x_j||^2 \le r_{ij}^2 \quad \text{for all } (i,j) \in \mathcal{E} \tag{5.30b}$$

$$\sum_{i=1}^{N} x_i = 0 \tag{5.30c}$$

Problem (5.30) is known as the Maximum Variance Unfolding problem, or MVU (Sun et al., 2006, Weinberger and Saul, 2006). Under the assumption that the communication graph $\mathcal{G}$ is connected, the MVU problem (5.30) has a (possibly not unique) optimal solution x^{opt}, which makes all the inequality constraints active (Sun et al., 2006). Notice the constraint (5.30c): its scope is to make the solution of (5.30) finite (in fact, without it we could take all the x_i to be the same and to be arbitrarily large).

Although the MVU problem (5.30) is non-convex, it is well-known that it can be transformed into a convex problem by the change of variables $X = xx^\top$. Pursuing this transformation we arrive at the convex SDP (Sun et al., 2006, Göring et al., 2008)

$$\underset{X}{\textbf{maximize}} \quad \text{trace}\,(X) \tag{5.31a}$$

$$\textbf{subject to} \quad X_{ii} + X_{jj} - X_{ij} - X_{ji} \le r_{ij}^2, \quad \text{for all } (i,j) \in \mathcal{E} \tag{5.31b}$$

$$\mathbf{1}_N^\top X \mathbf{1}_N = 0, \quad X \succeq 0 \tag{5.31c}$$

The non-convex problem (5.30) and convex (5.31) are equivalent in the sense that they yield the same optimal value, i.e., $x^{\text{opt}\top} x^{\text{opt}} = \text{trace}\,(X^{\text{opt}})$. For this reason, in many applications it is often convenient to transform the non-convex (5.30) into the convex (5.31), which can be solved efficiently (Weinberger and Saul, 2006, Weinberger et al., 2007).

Both the non-convex problem (5.30) and the convex one (5.31) can be solved with centralized algorithms (although, for the non-convex case the algorithms may lead to local optima). Furthermore, methods have been proposed to approximate the convex problem (5.31) in order to reduce its computational complexity when the size of the problem becomes large (e.g., $N > 1000$), see (Weinberger et al., 2007).

Distributed Solutions

Solving the two problems (5.30) and (5.31) in a distributed way is more challenging. In principle, the non-convex (5.30) could be solved using a Sequential Quadratic Programming approach (Bertsekas, 1982, 1999), and the resulting quadratic programs distributed among the nodes with the saddle-point iterations of Section 5.2. In practice, the validity of such a scheme is still in doubt, since the regularization parameters could endanger convergence, and even in the best case, the algorithm might lead to a local optimum. On the other hand, the convex formulation (5.31) is constrained via matrix (in)equalities that are difficult to decouple among the nodes. In particular, the projection over the constraint $X \succeq 0$ would require the knowledge of the spectral decomposition of the full matrix X, see (Boyd and Vandenberghe, 2004). The available distributed algorithm (Kempe and McSherry, 2008) to obtain such decomposition is limited to matrices that own the same sparsity of the underlying graph and therefore it is not directly applicable in this case. Furthermore, the knowledge of X^{opt} would not automatically imply that x^{opt} can be computed in a distributed way.

Nonetheless, we will see in the next sections how the dualization of the convex problem (5.31) overcomes most of the issues and allows us to devise globally optimal distributed algorithms.

5.3.2 The Dual of the MVU Problem and the Fastest Mixing Markov Problem

We describe in this section a rather well-known relationship between the MVU problem (5.30) and the Fastest Mixing Markov Process problem (Sun et al., 2006). In particular, we show that these two problems are the dual of one another. This link will be used in our proposed algorithm.

Consider the weighted and undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where we assign to each edge a weight $w_{ij} \in \mathbb{R}_+$. Let E the cardinality of $\mathcal{E}$ and let $w \in \mathbb{R}_+^E$ be the stacked vector of the weights, and let L be the laplacian of $\mathcal{G}$. The second smallest eigenvalue of the graph depends on the weights, which we indicate with the notation $\lambda_2(w)$. Finding the Fastest Mixing Markov Process on a graph, or FMMP, is the problem of determining the weights w that maximize the algebraic connectivity of the graph, under a certain linear bound on w. The FMMP problem can be written as

$$\underset{w}{\textbf{maximize}} \quad \lambda_2(w) \tag{5.32a}$$

$$\textbf{subject to} \quad \sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij} \leq 1 \tag{5.32b}$$

$$w \geq 0 \tag{5.32c}$$

or, since both the objective $\lambda_2(w)$ and the constraint function $\sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij}$ are positive homogeneous, in the equivalent (re-scaled) form (Sun et al., 2006)

$$\begin{aligned} \underset{w}{\textbf{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij} && \text{(5.33a)}\\ \textbf{subject to} \quad & \lambda_2(w) \geq 1 && \text{(5.33b)}\\ & w \geq 0 && \text{(5.33c)} \end{aligned}$$

where the inequality $w \geq 0$ has to be interpreted element-wise. The problems (5.32) and (5.33) are convex, since $\lambda(w)$ is a concave function of the weights w (De Gennaro and Jadbabaie, 2006). Moreover, we remark that at optimality $\lambda_2(w^{\text{opt}}) = 1$ for problem (5.33) (Sun et al., 2006).

Based on Lemma 4.1 of Chapter 3, problem (5.33) can be written as the SDP

$$\begin{aligned} \underset{w}{\textbf{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij} && \text{(5.34a)}\\ \textbf{subject to} \quad & L(w) + (1/N)\mathbf{1}_N\mathbf{1}_N^\top \succeq I_N && \text{(5.34b)}\\ & w \geq 0 && \text{(5.34c)} \end{aligned}$$

where, as usual, the Laplacian L depends linearly on the weights w.

Remark 5.1 *Notice that we can substitute the constraint $\lambda_2(w) \geq 1$ with $\lambda_2(w) \geq \bar{\lambda}_2$ for any positive $\bar{\lambda}_2 > 0$ without difficulty. In fact, due to the linearity of the Laplacian L on w, the scaling $\tilde{w} = w/\bar{\lambda}_2$ would normalize the problem to $\bar{\lambda}_2 = 1$.*

The dual of problem (5.34) is the convex problem

$$\begin{aligned} \underset{\tilde{X}}{\textbf{maximize}} \quad & \operatorname{trace}\left((I_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\tilde{X}\right) && \text{(5.35a)}\\ \textbf{subject to} \quad & \tilde{X}_{ii} + \tilde{X}_{jj} - \tilde{X}_{ij} - \tilde{X}_{ji} \leq r_{ij}^2, \text{ for all } (i,j)\in\mathcal{E} && \text{(5.35b)}\\ & \tilde{X} \succeq 0 && \text{(5.35c)} \end{aligned}$$

which, with the substitution of variable

$$X = (I_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)\tilde{X}(I_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top)$$

can be written as (Sun et al., 2006)

$$\begin{aligned} \underset{X}{\textbf{maximize}} \quad & \operatorname{trace}(X) && \text{(5.36a)}\\ \textbf{subject to} \quad & X_{ii} + X_{jj} - X_{ij} - X_{ji} \leq r_{ij}^2, \text{ for all } (i,j)\in\mathcal{E}, j > i && \text{(5.36b)}\\ & \mathbf{1}_N^\top X \mathbf{1}_N = 0, \quad X \succeq 0 && \text{(5.36c)} \end{aligned}$$

Problem (5.36) is equivalent to the convex MVU problem (5.31), thus the primal-dual relationship between FMMP problem and MVU problem. In particular, assuming that

Assumption 5.5 *There exists a Slater vector for problem* (5.31) *and problem* (5.33).

we can prove that the duality gap is zero, (Sun et al., 2006), meaning

$$\sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij}^{\mathrm{opt}} = \mathrm{trace}\left(X^{\mathrm{opt}}\right)$$

Furthermore when solving the convex MVU problem (5.31) at optimality, its dual variables will be optimal for the FMMP problem (5.34) and vice versa. We remark that this is also true for the non-convex MVU problem (5.30) if its global maximum is found. We formalize these primal-dual relationships by the use of KKT optimal conditions. In particular, the optimal couple $(w^{\mathrm{opt}}, X^{\mathrm{opt}})$ satisfies:

Primal-Dual feasibility:

$$w^{\mathrm{opt}} \geq 0, \quad L(w^{\mathrm{opt}}) \succeq I_N - (1/N)\mathbf{1}_N\mathbf{1}_N^\top$$

$$\mathbf{1}_N^\top X^{\mathrm{opt}}\mathbf{1}_N = 0, \quad X^{\mathrm{opt}} \succeq 0, \quad X_{ii}^{\mathrm{opt}} + X_{jj}^{\mathrm{opt}} - X_{ij}^{\mathrm{opt}} - X_{ji}^{\mathrm{opt}} \leq r_{ij}^2, \text{for all } (i,j) \in \mathcal{E}$$

Complementary slackness on edges:

$$(X_{ii}^{\mathrm{opt}} + X_{jj}^{\mathrm{opt}} - X_{ij}^{\mathrm{opt}} - X_{ji}^{\mathrm{opt}} - r_{ij}^2)w_{ij}^{\mathrm{opt}} = 0, \quad \text{for all } (i,j) \in \mathcal{E}$$

Matrix complementary slackness:

$$L(w^{\mathrm{opt}})X^{\mathrm{opt}} = X^{\mathrm{opt}}$$

The last condition means that the range of X^{opt} lies in the eigenspace of $L(w^{\mathrm{opt}})$ associated with $\lambda_2(w^{\mathrm{opt}})$ (which we recall to be one). This leads to the following result. Indicate with $\mathbf{v}_2^{\mathrm{opt}}$ the normalized eigenvector associated with $\lambda_2(w^{\mathrm{opt}})$ and call c^{opt} the optimal cost for the FMMP problem, i.e., $c^{\mathrm{opt}} = \sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij}$, then an optimal solution for X is

$$X^{\mathrm{opt}} = c^{\mathrm{opt}}\mathbf{v}_2^{\mathrm{opt}}{\mathbf{v}_2^{\mathrm{opt}}}^\top \tag{5.37}$$

Furthermore if $\lambda_2(w^{\mathrm{opt}})$ is isolated, this solution is also unique (Sun et al., 2006). The relation (5.37) also yields to

$$x^{\mathrm{opt}} = \sqrt{c^{\mathrm{opt}}}\mathbf{v}_2^{\mathrm{opt}} \tag{5.38}$$

for a global optimal optimizer of the non-convex problem (5.30).

Equation (5.38) will be an important ingredient in the design of a distributed global optimal algorithm for the non-convex problem (5.30) as we illustrate next.

5.3.3 Proposed Distributed Algorithm

We are interested in solving the MVU problem (5.30) in a distributed fashion. In order to do so, we propose to utilize its dual convex FMMP (5.33) problem and a primal-dual subgradient technique. (We remark that $\lambda_2(w)$ is a non-smooth function of w). The intention is to design a global optimal distributed algorithm for (5.30) by solving in a distributed way the convex FMMP (5.33).

Preliminaries

First of all we recall some useful definitions and some needed preliminary results. Let f be a convex function $f : \mathbb{R}^n \to \mathbb{R}$, while let g be a concave function $g : \mathbb{R}^n \to \mathbb{R}$. The vector $\mathbf{s} \in \mathbb{R}^n$ is a subgradient of the convex function f at a point $x \in \mathbb{R}^n$ if

$$f(y) \geq f(x) + \langle \mathbf{s}, y - x \rangle, \quad \text{for all } y \in \mathbb{R}^n$$

while $\mathbf{s}$ is a subgradient[7] of the concave function g at a point $x \in \mathbb{R}^n$ if

$$g(y) \leq g(x) + \langle \mathbf{s}, y - x \rangle, \quad \text{for all } y \in \mathbb{R}^n$$

Given a strictly positive scalar $\varepsilon > 0$, the vector $\mathbf{s}$ is an ε-subgradient of f at a point $x \in \mathbb{R}^n$ if

$$f(y) \geq f(x) + \langle \mathbf{s}, y - x \rangle - \varepsilon, \quad \text{for all } y \in \mathbb{R}^n$$

while $\mathbf{s}$ is an ε-subgradient of g at a point $x \in \mathbb{R}^n$ if

$$g(y) \leq g(x) + \langle \mathbf{s}, y - x \rangle + \varepsilon, \quad \text{for all } y \in \mathbb{R}^n$$

The concept of ε-subgradient is useful when the computation of the subgradient of a given function is affected by approximation errors. This is the reason why the schemes that employ ε-subgradients instead of subgradients are often referred to as approximate subgradient methods (Kiwiel, 2004).

Consider the convex FMMP problem (5.33). Let the vector $\mathbf{q_{ij}} \in \mathbb{R}^N$ be

$$(\mathbf{q_{ij}})_q = \begin{cases} 1 & \text{if } q = i \\ -1 & \text{if } q = j \\ 0 & \text{otherwise} \end{cases} \tag{5.39}$$

We remark that the subscripts $\mathbf{i}$ and $\mathbf{j}$ are bold since the object $\mathbf{q_{ij}}$ is a vector that refers to the nodes i and j and not the element (i, j) of $\mathbf{q}$.

The Laplacian $L(w)$ of the underlying graph $\mathcal{G}$ on which problem (5.33) is based upon can be written as

$$L(w) = \sum_{(i,j)\in\mathcal{E}} \mathbf{q_{ij}}\mathbf{q_{ij}}^\top w_{ij}$$

The algebraic connectivity of $L(w)$ is a concave function of w since for every $\tilde{w} \in \mathbb{R}^E_+$ (De Gennaro and Jadbabaie, 2006)

$$\lambda_2(\tilde{w}) \leq \lambda_2(w) + \text{trace}\left(\langle \mathbf{v}_2\mathbf{v}_2^\top, L(\tilde{w}) - L(w)\rangle\right)$$

where $\mathbf{v}_2$ is the eigenvector associated to the second smallest eigenvalue of $L(w)$. Substituting the expression of $L(w)$ we obtain

$$\lambda_2(\tilde{w}) \quad \leq \quad \lambda_2(w) + \text{trace}\left(\langle \mathbf{v}_2\mathbf{v}_2^\top, \sum_{(i,j)\in\mathcal{E}} \mathbf{q_{ij}}\mathbf{q_{ij}}^\top(\tilde{w}_{ij} - w_{ij})\rangle\right) =$$

[7] We note that some authors refer to $\mathbf{s}$ as supergradient if it is the subgradient of a concave function.

$$
\begin{aligned}
&\lambda_2(w) + \text{trace}\left(\sum_{(i,j)\in\mathcal{E}} \mathbf{v}_2\mathbf{v}_2^\top \mathbf{q_{ij}}\mathbf{q_{ij}}^\top(\tilde{w}_{ij} - w_{ij})\right) = \\
&\lambda_2(w) + \sum_{(i,j)\in\mathcal{E}} \text{trace}\left(\mathbf{v}_2\mathbf{v}_2^\top \mathbf{q_{ij}}\mathbf{q_{ij}}^\top\right)(\tilde{w}_{ij} - w_{ij}) = \\
&\lambda_2(w) + \sum_{(i,j)\in\mathcal{E}} (\mathbf{v}_{2i} - \mathbf{v}_{2j})^2(\tilde{w}_{ij} - w_{ij})
\end{aligned}
$$

where $\mathbf{v}_{2i}$ and $\mathbf{v}_{2j}$ are the i-th and j-th component of $\mathbf{v}_2$. Therefore the vector $\nabla_w\lambda_2(w) \in \mathbb{R}^E$ with components

$$
(\nabla_w\lambda_2(w))_{ij} = (\mathbf{v}_{2i} - \mathbf{v}_{2j})^2 \tag{5.40}
$$

is a subgradient for (the concave) $\lambda_2(w)$ at w.

Let $\mathcal{L}(w,\mu)$ be the Lagrangian function associated with the FMMP problem (5.33), i.e.,

$$
\mathcal{L}(w,\mu) = \sum_{(i,j)\in\mathcal{E}} r_{ij}^2 w_{ij} + \mu(1 - \lambda_2(w)) \tag{5.41}
$$

where $\mu \in \mathbb{R}_+$ is the dual variable of w. Let $\nabla_w\mathcal{L}(w,\mu)$ and $\nabla_\mu\mathcal{L}(w,\mu)$ be the subgradients of $\mathcal{L}(w,\mu)$ with respect to w and μ, respectively. These subgradients can be expressed component-wise as

$$
\begin{aligned}
(\nabla_w\mathcal{L}(w,\mu))_{ij} &= r_{ij}^2 - \mu\,(\nabla_w\lambda_2(w))_{ij} = r_{ij}^2 - \mu(\mathbf{v}_{2i} - \mathbf{v}_{2j})^2 \\
\nabla_\mu\mathcal{L}(w,\mu) &= 1 - \lambda_2(w).
\end{aligned}
$$

We note that, for Assumption 5.5 and the existence of a solution for problem (5.33), there exist two closed, convex, and bounded set $\mathbb{W} \subset \mathbb{R}_+$ and $\mathbb{M} \subset \mathbb{R}_+$ so that $w \in \mathbb{W}$ and $\mu \in \mathbb{M}$. These sets are computable a priori[8] (Nedić and Ozdaglar, 2009).

From the existence of these two sets, the following standard assumption holds true for our problem formulation.

Assumption 5.6 *The subgradients $\nabla_w\mathcal{L}(w,\mu)$ and $\nabla_\mu\mathcal{L}(w,\mu)$ are uniformly bounded, i.e., there is a constant $\Lambda > 0$ such that*

$$
\|\nabla_w\mathcal{L}(w,\mu)\| \le \Lambda, \quad \|\nabla_\mu\mathcal{L}(w,\mu)\| \le \Lambda \quad \textit{for all } w \in \mathbb{W}, \mu \in \mathbb{M}
$$

Primal-dual iterations

In order to solve in a distributed way the FMMP problem (5.33), we consider the primal-dual iterations

$$
w^{(\tau+1)} = \mathcal{P}_{\mathbb{W}}\left[w^{(\tau)} - \alpha\nabla_w\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right] \tag{5.42}
$$

$$
\mu^{(\tau+1)} = \mathcal{P}_{\mathbb{M}}\left[\mu^{(\tau)} + \alpha\nabla_\mu\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right] \tag{5.43}
$$

[8] As a matter of fact, in practical situations, we could assume $w \in \mathbb{R}_+$ and $\mu \in \mathbb{R}_+$.

with a constant step size $\alpha > 0$.

We define the running averages $\bar{w}^{(\tau)}$ and $\bar{\mu}^{(\tau)}$ generated by:

$$\bar{w}^{(\tau)} = \frac{1}{\tau}\sum_{j=0}^{\tau-1} w^{(j)}, \quad \bar{\mu}^{(\tau)} = \frac{1}{\tau}\sum_{j=0}^{\tau-1} \mu^{(j)}$$

We can cite the following theorem from (Nedić and Ozdaglar, 2009) that guarantees convergence of the couple $(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)})$ to a saddle-point of the Lagrangian (5.41).

Theorem 5.2 [Proposition 1 of Nedić and Ozdaglar (2009)] *Under Assumption 5.6 the following relations for the iterates* (5.42)-(5.43) *hold true:*

(a) For all $\tau \geq 1$,

$$-\frac{||\mu^{(0)} - \mu^{\text{opt}}||^2}{2\alpha\tau} - \frac{\alpha\Lambda^2}{2} \leq \frac{1}{\tau}\sum_{j=0}^{\tau-1} \mathcal{L}(w^{(j)}, \mu^{(j)}) - \mathcal{L}(w^{\text{opt}}, \mu^{\text{opt}}) \leq \frac{||w^{(0)} - w^{\text{opt}}||^2}{2\alpha\tau} + \frac{\alpha\Lambda^2}{2}$$

(b) The averages $\bar{w}^{(\tau)}$ *and* $\bar{\mu}^{(\tau)}$ *satisfy the following relation for all* $\tau \geq 1$*:*

$$-\frac{||\mu^{(0)} - \mu^{\text{opt}}||^2 + ||w^{(0)} - \bar{w}^{(\tau)}||^2}{2\alpha\tau} - \alpha\Lambda^2 \leq \mathcal{L}(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)}) - \mathcal{L}(w^{\text{opt}}, \mu^{\text{opt}}) \leq \frac{||w^{(0)} - w^{\text{opt}}||^2 + ||\mu^{(0)} - \bar{\mu}^{(\tau)}||^2}{2\alpha\tau} + \alpha\Lambda^2$$

The result in part *(a)* provides bounds on the averaged function values

$$\frac{1}{\tau}\sum_{j=0}^{\tau-1} \mathcal{L}(w^{(j)}, \mu^{(j)})$$

in terms of the distances of the initial iterates $w^{(0)}$ and $\mu^{(0)}$ from the vectors w^{opt} and μ^{opt} that constitute a saddle point of $\mathcal{L}(w, \mu)$. In particular, the averaged function values converge to the saddle point value $\mathcal{L}(w^{\text{opt}}, \mu^{\text{opt}})$ within error level $\alpha\Lambda^2/2$. This convergence goes as $1/\tau$ with the number of iteration[9]. The result in part *(b)* gives bounds on the function value $\mathcal{L}(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)})$ of the averaged iterates $\bar{w}^{(\tau)}$ and $\bar{\mu}^{(\tau)}$ in terms of the distances of the averaged iterates from the initial iterates and saddle point vectors. Under the assumption that the iterates generated by the subgradient algorithm (5.42)-(5.43) are bounded, this result shows that the function values of the averaged iterates $\mathcal{L}(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)})$ converge to the saddle-point value $(w^{\text{opt}}, \mu^{\text{opt}})$ within error level $\alpha\Lambda^2$ as $1/\tau$. The error level is due to the use of a constant step size and can be controlled by choosing a smaller

[9]In practice we could say that the rate of convergence is $1/\tau$ (Koshal et al., 2011).

step size value at the price of increasing the number of iterations τ. Therefore, Theorem 5.2 provides explicit tradeoffs between accuracy (in terms of α) and computational complexity (in terms of τ) in choosing the step size value.

In the following theorem we characterize the value of the optimal dual variable μ^{opt}, which will enable us to derive a global optimal optimizer for the non-convex MVU (5.30).

Theorem 5.3 *The optimal value of the dual variable μ^{opt} is unique and equal to the cost of the FMMP problem* (5.33)*, i.e., $\mu^{\text{opt}} = c^{\text{opt}}$.*

Proof. An optimal point for the FMMP problem (5.33) is a fixed point of the iterations (5.42)-(5.43), due to Theorem 5.2. In particular, an optimizer of (5.33), must satisfy

$$w_{ij}^{\text{opt}} = \mathcal{P}_{\mathbb{W}}\left[w_{ij}^{\text{opt}} - \alpha(r_{ij}^2 - \mu^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2)\right], \quad \text{for all } (i,j) \in \mathcal{E}.$$

Since $\alpha > 0$, and since $w_{ij}^{\text{opt}} < \max_{w_{ij}} ||\mathbb{W}||$, this means that: either $w_{ij}^{\text{opt}} = 0$ and $r_{ij}^2 - \mu^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2 > 0$ or $w_{ij}^{\text{opt}} > 0$ and

$$r_{ij}^2 - \mu^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2 = 0.$$

Due to the fact that w^{opt} is not null, we can write

$$\sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}}\left(r_{ij}^2 - \mu^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2\right) = 0,$$

thus

$$\sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}} r_{ij}^2 = \mu^{\text{opt}} \sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2,$$

and therefore

$$\begin{aligned}
c^{\text{opt}} &= \mu^{\text{opt}} \sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}}(\mathbf{v}_{2i}^{\text{opt}} - \mathbf{v}_{2j}^{\text{opt}})^2 \\
&= \mu^{\text{opt}} \sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}} \text{trace}\left(\mathbf{v}_2^{\text{opt}}\mathbf{v}_2^{\text{opt}\top}\mathbf{q_{ij}}\mathbf{q_{ij}}^\top\right) \\
&= \mu^{\text{opt}} \text{trace}\left(\mathbf{v}_2^{\text{opt}}\mathbf{v}_2^{\text{opt}\top} \sum_{(i,j)\in\mathcal{E}} w_{ij}^{\text{opt}}\mathbf{q_{ij}}\mathbf{q_{ij}}^\top\right) \\
&= \mu^{\text{opt}} \text{trace}\left(\mathbf{v}_2^{\text{opt}}\mathbf{v}_2^{\text{opt}\top} L(w^{\text{opt}})\right) \\
&= \mu^{\text{opt}} \text{trace}\left(\mathbf{v}_2^{\text{opt}}\mathbf{v}_2^{\text{opt}\top}\right) = \mu^{\text{opt}}
\end{aligned}$$

where we use the fact that $\mathbf{v}_2^{\text{opt}\top} L(w^{\text{opt}}) = \mathbf{v}_2^{\text{opt}\top}$, since $\lambda_2(w^{\text{opt}}) = 1$. □

Theorems 5.2 and 5.3 with the iterations (5.42)-(5.43) provide a way to compute an optimal solution for the FMMP problem (5.33), as well as for the non-convex MVU problem (5.30). In fact, recalling that by equation (5.38), $x^{\text{opt}} = \sqrt{c^{\text{opt}}}\mathbf{v}_2^{\text{opt}}$, once the couple $(\mu^{\text{opt}}, \mathbf{v}_2^{\text{opt}})$ is available through the iterations (5.42)-(5.43), due to the equivalence $\mu^{\text{opt}} = c^{\text{opt}}$, one can readily compute x^{opt}.

However, the iterations (5.42)-(5.43) are not distributed, since they require the knowledge of the algebraic connectivity and its associated eigenvector.

Approximate Distributed Solution

In this section we propose a way to distribute the computation of the subgradients of $\mathcal{L}(w, \mu)$. Furthermore, we will analyze the case in which these subgradients are affected by some approximation error. This case is of practical importance when the communication effort among the nodes has to be limited, and therefore the iterative distributed algorithm to compute the subgradients of $\mathcal{L}(w, \mu)$ has to be stopped before reaching convergence.

Since $L(w)$ is a sparse matrix, we can now utilize the already mentioned distributed algorithm (Kempe and McSherry, 2008) to compute its eigenvalues and eigenvector. This technique, named by the authors as DOI algorithm (for decentralized orthogonal iteration), computes the spectral decomposition of a matrix M that owns the same sparsity of the underlying graph $\mathcal{G}$. The method is based on a QR decomposition and a consensus iteration and converges within an accuracy of ε to the eigenspace of the matrix $M \in \mathbb{R}^{N\times N}$ in $O(\log^2(N/\varepsilon)1/\lambda_2(\mathcal{G}))$ rounds of communication/computations. Using this method, each node of the network has a copy of $\mathbf{v}_2$ with which they can compute locally the algebraic connectivity (by the multiplication of $\mathbf{v}_2$ with their row of the Laplacian).

The iterations (5.42)-(5.43) with the DOI algorithm could be used to solve the FMMP/MVU problems in a distributed way. It remains to prove that the convergence result of Theorem 5.2 still holds if the subgradients of the Lagrangian function (i.e., the algebraic connectivity and its associated eigenvector) are computed up to a prescribed accuracy ε.

Let $\nabla_{w,\varepsilon}\mathcal{L}(w, \mu)$ and $\nabla_{\mu,\varepsilon}\mathcal{L}(w, \mu)$ be ε-subgradients of $\mathcal{L}(w, \mu)$ with respect to w and μ, respectively. Consider the modification of the iterates (5.42)-(5.43) as

$$\begin{aligned} w^{(\tau+1)} &= \mathcal{P}_{\mathbb{W}}\left[w^{(\tau)} - \alpha\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)}, \mu^{(\tau)})\right] && (5.44)\\ \mu^{(\tau+1)} &= \mathcal{P}_{\mathbb{M}}\left[\mu^{(\tau)} + \alpha\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)}, \mu^{(\tau)})\right] && (5.45)\end{aligned}$$

Moreover, consider the modification of the Assumption 5.6 as

Assumption 5.7 *The ε-subgradients $\nabla_{w,\varepsilon}\mathcal{L}(w, \mu)$ and $\nabla_{\mu,\varepsilon}\mathcal{L}(w, \mu)$ are uniformly bounded, i.e., there is a constant $\Lambda_\varepsilon > 0$ such that*

$$\|\nabla_{w,\varepsilon}\mathcal{L}(w, \mu)\| \le \Lambda_\varepsilon, \quad \|\nabla_{\mu,\varepsilon}\mathcal{L}(w, \mu)\| \le \Lambda_\varepsilon \quad \textit{for all } w \in \mathbb{W}, \mu \in \mathbb{M}$$

We formalize the convergence of the running averages $\bar{w}^{(\tau)}$ and $\bar{\mu}^{(\tau)}$ based on the approximated iterates (5.44)-(5.45) to a saddle-point of $\mathcal{L}(w, \mu)$ in the following theorem.

Theorem 5.4 *Under Assumption 5.7 the following relations for the iterates (5.44)-(5.45) hold true:*

(a) For all $\tau \geq 1$,

$$-\frac{||\mu^{(0)}-\mu^{\text{opt}}||^2}{2\alpha\tau}-\frac{\alpha\bar{\Lambda}_\varepsilon^2}{2} \leq \frac{1}{\tau}\sum_{j=0}^{\tau-1}\mathcal{L}(w^{(j)},\mu^{(j)})-\mathcal{L}(w^{\text{opt}},\mu^{\text{opt}}) \leq \frac{||w^{(0)}-w^{\text{opt}}||^2}{2\alpha\tau}+\frac{\alpha\bar{\Lambda}_\varepsilon^2}{2}$$

(b) The averages $\bar{w}^{(\tau)}$ *and* $\bar{\mu}^{(\tau)}$ *satisfy the following relation for all* $\tau \geq 1$*:*

$$-\frac{||\mu^{(0)}-\mu^{\text{opt}}||^2+||w^{(0)}-\bar{w}^{(\tau)}||^2}{2\alpha\tau}-\alpha\bar{\Lambda}_\varepsilon^2 \leq \mathcal{L}(\bar{w}^{(\tau)},\bar{\mu}^{(\tau)})-\mathcal{L}(w^{\text{opt}},\mu^{\text{opt}}) \leq \frac{||w^{(0)}-w^{\text{opt}}||^2+||\mu^{(0)}-\bar{\mu}^{(\tau)}||^2}{2\alpha\tau}+\alpha\bar{\Lambda}_\varepsilon^2$$

with $\bar{\Lambda}_\varepsilon^2 = \Lambda_\varepsilon^2 + 2\varepsilon/\alpha$.

Proof. The proof follows from Proposition 1 of (Nedić and Ozdaglar, 2009). First we prove that:

$$||w^{(\tau+1)}-w||^2 \leq ||w^{(\tau)}-w||^2 - 2\alpha(\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w,\mu^{(\tau)}))+\alpha^2\bar{\Lambda}_\varepsilon^2 \tag{5.46}$$

$$||\mu^{(\tau+1)}-\mu||^2 \leq ||\mu^{(\tau)}-\mu||^2 + 2\alpha(\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w^{(\tau)},\mu))+\alpha^2\bar{\Lambda}_\varepsilon^2 \tag{5.47}$$

In order to show (5.46), we can expand $||w^{(\tau+1)}-w||^2$ into

$$\begin{aligned} ||w^{(\tau+1)}-w||^2 &= \left\|\mathcal{P}_{\mathbb{W}}\left[w^{(\tau)}-\alpha\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right]-w\right\|^2 \\ &\leq ||w^{(\tau)}-w||^2 - 2\alpha\langle\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),w^{(\tau)}-w\rangle+\Lambda_\varepsilon^2 \end{aligned}$$

where we use the non-expansivity property of the projection. By the definition of ε-subgradient and since the function $\mathcal{L}(w,\mu)$ is convex in w we have

$$\begin{aligned} \langle\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),w-w^{(\tau)}\rangle-\varepsilon &\leq \mathcal{L}(w,\mu^{(\tau)})-\mathcal{L}(w^{(\tau)},\mu^{(\tau)}) \\ -\langle\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),w^{(\tau)}-w\rangle &\leq -(\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w,\mu^{(\tau)}))+\varepsilon \end{aligned}$$

and therefore

$$||w^{(\tau+1)}-w||^2 \leq ||w^{(\tau)}-w||^2 - 2\alpha(\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w,\mu^{(\tau)}))+\alpha^2\Lambda_\varepsilon^2+2\alpha\varepsilon$$

which is (5.46).

In order to show (5.47), we can expand $||\mu^{(\tau+1)}-\mu||^2$ into

$$\begin{aligned} ||\mu^{(\tau+1)}-\mu||^2 &= \left\|\mathcal{P}_{\mathbb{M}}\left[\mu^{(\tau)}+\alpha\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right]-\mu\right\|^2 \\ &\leq ||\mu^{(\tau)}-\mu||^2 + 2\alpha\langle\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),\mu^{(\tau)}-\mu\rangle+\Lambda_\varepsilon^2 \end{aligned}$$

where we use the non-expansivity property of the projection. By the definition of ε-

subgradient and since the function $\mathcal{L}(w,\mu)$ is concave in μ we have

$$\begin{aligned}\langle\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),\mu-\mu^{(\tau)}\rangle+\varepsilon &\geq \mathcal{L}(w^{(\tau)},\mu)-\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\\ \langle\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)}),\mu^{(\tau)}-\mu\rangle &\leq (\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w^{(\tau)},\mu))+\varepsilon\end{aligned}$$

and therefore

$$||\mu^{(\tau+1)}-\mu||^2\leq||\mu^{(\tau)}-\mu||^2+2\alpha(\mathcal{L}(w^{(\tau)},\mu^{(\tau)})-\mathcal{L}(w^{(\tau)},\mu))+\alpha^2\Lambda_\varepsilon^2+2\alpha\varepsilon$$

which is (5.47).

The proof of Proposition 1 of (Nedić and Ozdaglar, 2009), i.e., Theorem 5.2, is based on (5.46) and (5.47) with $\varepsilon=0$. Substituting Λ^2 with $\bar{\Lambda}_\varepsilon^2=\Lambda_\varepsilon^2+2\varepsilon/\alpha$ it is not difficult to see that the analysis in (Nedić and Ozdaglar, 2009) still holds and therefore the claims *(a)* and *(b)* also hold. □

Theorems 5.4 and 5.3 with the iterations (5.44)-(5.45) provide a way to compute an approximately optimal solution for the FMMP problem (5.33), as well as for the non-convex MVU problem (5.30) in a distributed way as summarized in Algorithm 5.2.

Algorithm 5.2 Primal-Dual Algorithm for the MVU and FMMP problems

```
1: Input $w^{(\tau)},\mu^{(\tau)}$
   ▷ Available data: $\alpha,\varepsilon,L(w),(r_{ij}|\text{for each}(i,j)\in\mathcal{E}),\mathbb{W},\mathbb{M}$
2: Determine $\lambda_2(w)$ and $\mathbf{v}_2$ of $L(w^{(\tau)})$ via the distributed DOI algorithm of Kempe and McSherry (2008) up
   to an accuracy of $\varepsilon$
3: Compute: $\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})$ and $\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})$ as
   $$\begin{aligned}(\nabla_{w,\varepsilon}\mathcal{L}(w,\mu))_{ij} &= r_{ij}^2-\mu\left(\nabla_w\lambda_2(w)\right)_{ij}=r_{ij}^2-\mu(\mathbf{v}_{2i}-\mathbf{v}_{2j})^2\\ \nabla_{\mu,\varepsilon}\mathcal{L}(w,\mu) &= 1-\lambda_2(w)\end{aligned}$$
4: Compute:
   $$\begin{aligned}w^{(\tau+1)} &= \mathcal{P}_{\mathbb{W}}\left[w^{(\tau)}-\alpha\nabla_{w,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right]\\ \mu^{(\tau+1)} &= \mathcal{P}_{\mathbb{M}}\left[\mu^{(\tau)}+\alpha\nabla_{\mu,\varepsilon}\mathcal{L}(w^{(\tau)},\mu^{(\tau)})\right]\end{aligned}$$
5: Compute the iteration of the MVU problem: $x^{(\tau)}=\sqrt{\mu^{(\tau)}}\mathbf{v}_2$
6: Output: $w^{(\tau+1)},\mu^{(\tau+1)},x^{(\tau)}$
```

In the next section we show a small numerical example to assess the performance of the proposed algorithm.

5.3.4 Numerical Example

We use a small numerical example from (Sun et al., 2006) to show the performance of the primal-dual iterations (5.44)-(5.45) with the DOI algorithm applied to the FMMP and MVU problems, i.e., (5.33) and (5.30). Let the r_{ij} be $r_{12}=1$, $r_{13}=2$, $r_{23}=1$, $r_{34}=1$, $r_{45}=1$, and $r_{46}=2$. Figure 5.2 gives a pictorial representation of the problem.

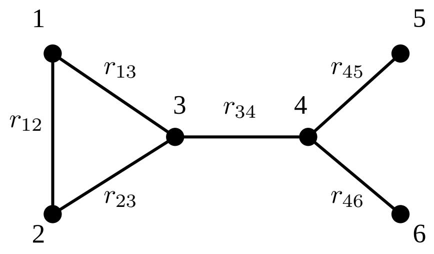


Figure 5.2: *Representation of the graph of the numerical example.*

The unique primal optimal solution (up to a multiplication by an orthogonal matrix) for the MVU problem (5.30) is

$$x_1^{\text{opt}} = -2.5, \quad x_2^{\text{opt}} = -1.5, \quad x_3^{\text{opt}} = -0.5, \quad x_4^{\text{opt}} = 0.5, \quad x_5^{\text{opt}} = 1.5, \quad x_6^{\text{opt}} = 2.5$$

while the optimal set of Lagrangian multipliers w_{ij} (thus the set of solutions of the dual FMMP problem (5.33)) is:

$$w_{12}^{\text{opt}} = u, \quad w_{23}^{\text{opt}} = 1.5 + u, \quad w_{13}^{\text{opt}} = 1.25 - 0.5u,$$

$$w_{34}^{\text{opt}} = 4.5, \quad w_{45}^{\text{opt}} = 1.5, \quad w_{46}^{\text{opt}} = 1.25$$

with the parameter $u \in [0, 2.5]$. The achieved optimal cost is $c^{\text{opt}} = 17.5$.

We use the iterations (5.44)-(5.45) with $\alpha = 0.15$ and $\varepsilon = 0.01$, starting from a random initial condition for $w^{(0)}$ and $\mu^{(0)}$. Figure 5.3 illustrates the convergence of $\mathcal{L}(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)})$ to a saddle point of $\mathcal{L}(w, \mu)$. We report that the total number of communication and computation iterations of the DOI algorithm, per iteration τ, was on average 12, and the computations required around 0.4 ms per node per iteration τ, on an Intel Core *i5* (2.3 GHz and 4GB DDR3) laptop. This leads to a total required time of 2 s if the scheme is run up to $\tau = 5000$. These communication/computation requirements are considered acceptable in many applications, especially in sensor network scenarios.

Finally we report the achieved tolerances

$$|\mathcal{L}(\bar{w}^{(5000)}, \bar{\mu}^{(5000)}) - \mathcal{L}(w^{\text{opt}}, \mu^{\text{opt}})| = 0.01, \quad |\lambda_2(\bar{w}^{(5000)}) - 1| = 0.03$$

which can be considered acceptable given the value of ε and α.

5.3.5 Extension to Multi-dimensional Problems and Localization Applications

In this section we extend the previous results to the case in which the variable x_i is a vector, meaning $x_i \in \mathbb{R}^n$, with $n > 1$. This scenario is particularly useful in localization problems (Weinberger et al., 2007). In order to see this, consider N different sensor nodes sparsely placed in an n dimensional space. Let $x_i \in \mathbb{R}^n$ be the position of sensor node i. Assume that each node can determine its distance to the closest neighboring nodes and let r_{ij} be this distance for each connected couple of nodes (i, j). The problem of determining all the positions of the nodes via the measurements r_{ij} is called (anchor-free) localization

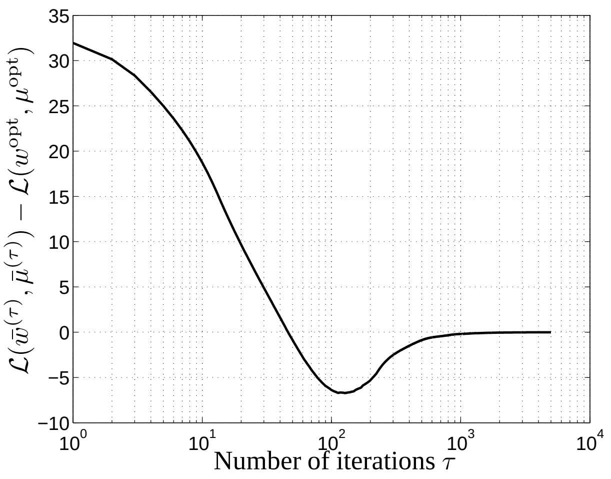


Figure 5.3: *Convergence of $\mathcal{L}(\bar{w}^{(\tau)}, \bar{\mu}^{(\tau)})$ to a saddle point of $\mathcal{L}(w, \mu)$ with respect to the iteration number τ.*

problem and it can be written as (Weinberger et al., 2007)

$$\underset{x_1,\dots,x_N}{\textbf{maximize}} \quad \sum_{i=1}^{N} ||x_i||^2 \tag{5.48a}$$

$$\textbf{subject to} \quad ||x_i - x_j||^2 = r_{ij}^2 \quad \text{for all } (i,j) \in \mathcal{E} \tag{5.48b}$$

$$\sum_{i=1}^{N} x_i = 0 \tag{5.48c}$$

Problem (5.48) is similar to the MVU problem (5.30) with equality constraints instead of inequalities, and via minor modifications (Sun et al., 2006) one can pass from one problem to the other.

To solve the MVU problem (5.30) for a multi-dimensional case (in a globally optimal way), we proceed in the same way as did in the scalar scenario. First, we define the matrix $X \in \mathbb{R}^{N\times N}$ as the Gramian matrix $X = xx^\top$, where $x = (x_1, \dots, x_N)^\top$. Then, we formulate the convex problem (5.31) and its dual the FMMP problem (5.33). We note that these problems are not affected by x_i not being a scalar. Therefore all the analysis of convergence of Algorithm 5.2 is still valid for $x_i \in \mathbb{R}^n$. The only notable difference is that for the multi-dimensional case the geometric multiplicity of $\lambda_2(w)$ is greater than one, in particular it is n (Sun et al., 2006). This implies that $\lambda_2 = \lambda_3 = \cdots = \lambda_{n+1} = 1$ and therefore, the optimal X will be written as

$$X^{\text{opt}} = \frac{c^{\text{opt}}}{n} \sum_{i=2}^{n+1} \mathbf{v}_i^{\text{opt}} \mathbf{v}_i^{\text{opt}\,\top} \tag{5.49}$$

while

$$x = \sqrt{\frac{c^{\mathrm{opt}}}{n}}(\mathbf{v}_2^{\mathrm{opt}}, \dots, \mathbf{v}_{n+1}^{\mathrm{opt}})^\top \tag{5.50}$$

With these relations we can compute the optimal value for x_i and solve the multi-dimensional MVU problem (5.30) in a distributed way via Algorithm 5.2.

We remark that the found solution will be globally optimal (since we are solving the dual convex problem). This is also true for the localization problem (5.48) (in case the underlying graph is rigid), if it is solved with Algorithm 5.2. This means that we are able to find the physical locations of the nodes up to rotations and reflections (more details on this problem for the centralized setting can be found in (Weinberger et al., 2007)).

5.4 Conclusions

In this chapter we have focused on certain classes of convex and non-convex networked optimization problems. We have shown how resource allocation constraints could be handled in a distributed iterative way keeping the number of computation/communication rounds for each node of the network reasonable for real application scenarios. Finally, we have illustrated how the proposed schemes could be used in realistic robotic network problems.

5.5 Open Problems and Future Work

There are a number of interesting open problems and future research directions for the work presented in this chapter, which are summarized in the following.

Convex resource allocation problems

The regularized primal-dual scheme of Section 5.2 can be optimized further with a more in-depth study on the optimal design of the step sizes and the regularization parameters. Furthermore, other approaches to speed up convergence, such as multi-step methods (Ghadimi et al., 2011), could be analyzed.

Finally, more extensive simulation studies on a variety of application scenarios should be performed to further assess the performance of the scheme in realistic situations.

Sequential Quadratic Programming approach

An alternative method to solve the non-convex MVU problem (5.30) in a distributed way is by standard Sequential Quadratic Programming. As pointed out in Section 5.3.1 this method could encounter convergence problems when the sequence of quadratic programs is regularized using the saddle-point algorithm of Section 5.2. In order to illustrate this, we recall the concept of the Sequential Quadratic Programming approach.

The Sequential Quadratic Programming approach, in short SQP approach, is based on a sequence of linearizations of the original non-convex problem. Let $x^{[\kappa]}$ be the value of the optimization variable x at the iteration step κ. Given an initial iterate $x^{[0]}$, the SQP method consists of the sequence:

- solve, at each $\kappa \geq 0$, the quadratic problem

$$\begin{aligned} \underset{\delta x}{\textbf{minimize}} \quad & -2{x^{[\kappa]}}^\top \delta x + \frac{1}{2}\nu \left\|\delta x\right\|^2 && \text{(5.51a)} \\ \textbf{subject to} \quad & 2(x_i^{[\kappa]} - x_j^{[\kappa]})^\top(\delta x_i - \delta x_j) + \\ & ||x_i^{[\kappa]} - x_j^{[\kappa]}||^2 - r_{ij}^2 \leq 0 \quad \text{for all } (i,j) \in \mathcal{E} && \text{(5.51b)} \\ & \mathbf{1}^\top \delta x = -\mathbf{1}^\top x^{[\kappa]} && \text{(5.51c)} \end{aligned}$$

  with optimization variable δx and where ν is a strictly positive scalar, i.e., $\nu > 0$. The optimizer of (5.51) is indicated with δx^{opt};

- update $x^{[\kappa+1]} = x^{[\kappa]} + \gamma^{[\kappa]}\delta x^{\mathrm{opt}}$, with a strictly positive scalar step-size $\gamma^{[\kappa]}$.

Conditions on the step size $\gamma^{[\kappa]}$ for the sequence of $x^{[\kappa]}$ to converge to a local maximum of the original non-convex MVU problem (5.30) can be found in the standard references (Bertsekas, 1982, 1999). We remark that ν in the quadratic problem (5.51) is important for the convergence properties of the SQP algorithm and, in general, can be unrelated to the second order derivative of the non-convex cost function, (Bertsekas, 1982).

We can see immediately that the quadratic problem (5.51) is a particular instance of the convex resource allocation problem (5.1) and could be regularized and solved using the iterations (5.6)-(5.7) in a distributed way.

However, employing this regularization could undermine the convergence proofs of Bertsekas (1982) and therefore question the whole procedure. More analyses are needed to generalize some of the theorems in (Bertsekas, 1982) to this regularized case. Furthermore, an estimate on how the scheme depends on the number of nodes N would be beneficial. In fact, it would be justifiable to apply this regularized SQP approach (which is delivering only local optimizers) only in particular circumstances and if it scales better with the network size N than the proposed global optimal one, i.e., iterations (5.44)-(5.45).

A Dispersion Problem for Robotic Networks

In Section 5.3.4 we have considered a numerical example where the nodes were fixed. This type of problem is typical in sensor network applications where the network structure is predetermined. Interesting open challenges arise when we let the nodes be mobile as in Chapter 3, and the weights depend on the position of the nodes. The FMMP problem (5.33) can then be rewritten as the following non-convex optimization problem

$$\begin{aligned} \underset{x(k)}{\textbf{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}(k)} r_{ij}^2 f_w(||x_i(k) - x_j(k)||^2) && \text{(5.52a)} \\ \textbf{subject to} \quad & \lambda_2(x(k)) > \bar{\lambda}_2 && \text{(5.52b)} \end{aligned}$$

where the decision variables are the robot locations $x(k)$, while $\bar{\lambda}_2$ is the prescribed level of connectivity and $f_w(\cdot)$ is a function that describes how the weights depend on the pair-wise distance between the robots, see Chapter 3 for details.

Problem (5.52) can be seen as the maximization of the dispersion (i.e., the distance among the robots) with the guarantee of the maintenance of a prescribed level of connectivity. This problem has been studied in the robotic literature (Howard et al., 2002, Cortés et al., 2004, Susan and Dubowsky, 2005, Arsie and Frazzoli, 2007, Hussein and Stipanovic, 2007, Dimarogonas and Kyriakopoulos, 2009); however no clear guarantees to obtain a prescribed level of connectivity are given.

Further studies to solve this problem in a distributed way are needed. We expect that one could apply the methods presented in Chapter 3, with some modifications. Moreover, it would be interesting to see whether two-step sequential approaches (De Gennaro and Jadbabaie, 2006) could be used here with the iterations (5.44)-(5.45) to deliver distributed optimizers with given bounds on their distance to the centralized solution.

For the interested reader, a detailed discussion of this problem and its possible solutions is presented in (Simonetto et al., 2012b).

Chapter 6

Conclusions and Recommendations

6.1 Summary and Conclusions

In this thesis we have proposed and analyzed solutions for distributed estimation, control, and optimization problems in robotic networks. The presented algorithms are more suitable to be implemented in real-time applications and show improved performance with respect to the available literature (in terms of estimation accuracy and real-time execution).

In Chapter 2, we have investigated distributed nonlinear state estimation problems and we have proposed a common framework in which to design distributed estimators. This novel framework allows the usage of different estimators on different sensor nodes which might be very convenient in practice to implement local filtering algorithms whose computational requirements match the devices' hardware. The proposed framework is based on a merging mechanism that combines the local estimates and their covariance matrices coming from the different sensor nodes. From an implementation perspective the proposed framework does not require extensive communication among the sensor nodes and it is implementable in real-time.

Within the proposed framework, we have designed novel versions of distributed Moving Horizon Estimators (Algorithm 2.2), Particle Filters (Algorithm 2.3), Unscented and Extended Kalman Filters (Algorithm 2.4). The proposed distributed Moving Horizon Estimator can incorporate convex constraints into the estimation problem and it is guaranteed to be stable and converging to an unbiased estimate for the state. This estimator extends the ideas of Farina et al. (2010) to nonlinear dynamical systems.

Algorithm 2.3 proposes a distributed version of Particle Filters applicable in sensor networks where measurements are taken locally and communicated via an information exchange network. This formulation of distributed Particle Filters is parametric, meaning that the a posteriori PDF is parametrized by the mean and covariance of the particle population. This is often convenient in practice for robotic scenarios and gives significant improvements compared to the results available in the literature (Chapter 2, Section 2.4). In particular, we observe an increase in accuracy of an order of magnitude with respect to similar available distributed algorithms. We show that this is due to the proposed merging mechanism and common framework.

In Chapter 3, we have analyzed how to distribute the high computational demand of Particle Filters on the multiple parallel cores of GPU-architectures. We have illustrated via simulation and experimental results that nonlinear filtering can be run efficiently yet still deliver accurate estimates. In particular, using over a million particles we have implemented the proposed distributed computation scheme on a robotic arm experimental setup that involves a visual servo feedback control loop with a sampling frequency of 100 Hz (Chapter 3, Section 4.2.7). This result is remarkable given the experimental setup (an educational platform equipped with an off-the-shelf webcam), the achieved estimation error (3 mm RMS for the the position estimation), and compared to similar efforts in the available literature. In particular, our implementation outperforms (often by orders of magnitude) the state-of-the-art implementations for number of particles, state dimension, and/or runtime. Moreover, our method competes for estimation quality with standard sequential Particle Filters that are however even hundreds time slower in high-number-of-particle settings. In addition, we have illustrated how some of the different user-tunable parameters affect the estimation performance and discussed scalability and portability of the proposed scheme.

In Chapter 4, we have investigated two distributed control problems for robotic networks. First, we have presented a distributed real-time implementable solution to the maximization of the algebraic connectivity of the communication graph of a robotic network. Our method can handle more realistic agent dynamics than the methods available in the literature, including second-order dynamical systems. The proposed distributed algorithm is proven to deliver feasible solutions in only one round of communication with the neighboring robots (Chapter 4, Theorem 4.2). In addition, the solution monotonically increases the cost function (Chapter 4, Theorem 4.4) and persistent feasibility is proven to be a property of the resulting optimization problem under standard assumptions (Chapter 4, Theorem 4.5). Finally, the proposed solution can be adjusted locally by each agent based on available resources, and this adjustment can be done using local relative sub-optimality measures. All these characteristics make the proposed distributed solution implementable and adjustable in real-time, which is an important requirement for realistic robotic networks.

In the second part of Chapter 4, we have extended the proposed solution for the maximization of the algebraic connectivity and proposed a distributed and non-iterative solution for the problem of collectively tracking multiple mobile targets using a robotic network, while maintaining a certain level of connectivity. As for the problem of the maximization of the algebraic connectivity, our distributed algorithm has been proven to deliver feasible solutions in only one round of communication (Chapter 4, Theorems 4.9, 4.10, and 4.11).

Simulation results have confirmed the efficacy of our distributed approaches and shown their practical applicability. For both problems the distributed solutions have similar behavior with respect to the centralized approximations and scale well with the number of agents.

The distributed solutions of Chapter 4 can be seen as complementary solutions of standard subgradient algorithms, see for example (De Gennaro and Jadbabaie, 2006). In subgradient algorithms we implicitly assume that the communication among the agents is somehow cheaper than the computation onboard. This translates in having communication extensive iterative algorithms, in which at each iteration, each agent has to evaluate only a subgradient of a certain function. Our proposed solution lies on the other side of the “communication-computation” trade-off spectrum. In fact, we assume that each robot can solve a rather complex convex optimization problem, while the communication among them is limited. Given the current availability of cheap onboard micro-processors that have reasonable computational power, we believe that mobile robotic networks could benefit more from solutions closer to our proposed one.

In Chapter 5, we have analyzed issues related to certain classes of convex and non-convex networked optimization problems involving resource allocation constraints. We have discussed how these problems are relevant for robotic network applications and we have proposed additional interesting scenarios (related to target-tracking, localization, and dispersion control). We have shown how resource allocation constraints could be handled in a distributed iterative way, keeping the number of computation/communication rounds for each node of the network reasonable for realistic applications.

In Section 5.2 we have discussed convex optimization problems with resource allocation constraints. We have analyzed regularized iterative saddle-point methods and shown how to *embed* the resource allocation constraint in the iterations. We have proposed a solution in Algorithm 5.1 and Theorem 5.1 that is faster than standard subgradient algorithms (in particular, it has a geometrical convergence rate). As a result, our algorithm is more suitable for real-time implementation in realistic robotic networks. The direction we have followed in this section offers another approach for real-time solutions to distributed optimization problem. While, in Chapter 4 we have focused on non-iterative methods that guarantee feasibility, in this section we have proposed fast iterative methods that regularize the original problem and provide a sub-optimal solution with given sub-optimality bounds.

In Section 5.3 we have dealt with a non-convex resource allocation optimization problem, known as the Maximum Variance Unfolding problem and its convex dual the Fastest Mixing Markov Process problem. Both optimization problems have been solved via the same primal-dual distributed subgradient algorithm (Algorithm 5.2) with guaranteed convergence, even in the case where errors are present in the computations of the subgradients (Theorem 5.4). These errors are common if we consider a limited (and fixed) number of communication rounds among the computing units. As a result, our proposed solution is more suitable to be implemented in real-time. The scheme has been demonstrated on relevant sensor network applications and we have discussed possible extensions to mobile robotic networks such as dispersion problems.

6.2 Recommendations for Future Work

A number of open research challenges and possible directions to tackle them are presented in this section. First of all, we will discuss some specific items, summarizing or complementing the ones analyzed in the main chapters of the thesis. Then, we will give recommendations on broader research possibilities.

Specific research questions arise from the problems and solutions we have presented in this Thesis. These questions lead to tangible research directions that could broaden the applicability of the proposed algorithms. The main challenges encompass the following.

- The solution of the dispersion problem using the methods presented in Chapter 5 (as discussed in Chapter 5, Section 5.5). In this scenario, we want to move the robots as far as possible from each other while maintaining a prescribed level of connectivity. Given a stacked vector of edge-weights at the discrete time k, i.e. $w(k)$, the problem can be formulated as a Fast Mixing Markov Process for each discrete time k, whose solutions are the optimal weights at the discrete time $k+1$, i.e., $w(k+1)$.

One challenge to be addressed is how to control the robots to achieve the prescribed value of communication weights $w(k+1)$. We note that, although potential function methods could be in principle used as in (De Gennaro and Jadbabaie, 2006) to drive the robots to an approximated vector of weights $\hat{w}(k+1)$, guarantees that the difference $||w(k+1) - \hat{w}(k+1)||$ stays below a given threshold are more difficult to obtain.

- Robust Multi-Target tracking (as discussed in Chapter 4, Section 4.5). This challenge is related to the fact that in the problem formulations of Chapter 4 (Problem 4.1 and Problem 4.2) each agent knows the position of the neighboring agents and targets. Considering that in practice this information is typically affected by measurement and estimation errors, we could extend the problem formulation to more realistic applications. As a result, the proposed distributed algorithms (Algorithm 4.1, and Algorithm 4.2) have to be redesigned to be robust to estimation errors.

- Constrained Consensus (as discussed in Chapter 2, Section 2.6). One of the main limitations of current consensus algorithms is their inability to handle generic (non-) convex constraints. Extending the consensus protocols to this case would increase their potential in many applications, such as distributed nonlinear estimation.

- Optimization of the merging mechanism (Chapter 2, Algorithm 2.1). We have illustrated that the merging scheme does not take into consideration the correlation among the local estimates. This may deliver, in some circumstances, "optimistic" covariance matrices in the sense of Bar-Shalom et al. (2001). An optimization of the merging mechanism (taking into account the correlation) could lead to better nonlinear estimation.

Besides these specific directions of investigation, there are a number of other (more high-level) research questions that need our consideration. As a matter of fact, distributed estimation, control, and optimization are a very dynamic and active research fields that have still many unsolved fundamental questions. Among them, driven by the results and the focus of this Thesis, we highlight the following research problems.

Distributed algorithms for convex optimization problems implementable in real-time

As we have discussed in this thesis, one of the limitations of current state-of-the-art distributed optimization algorithms is that they require iterative schemes that guarantee feasibility of the solution with respect to the constraints often only asymptotically (i.e., assuming an infinite number of iterations). This is impractical for the real-time implementations of optimization-based controllers and estimators in robotic networks.

We have shown that real-time feasible (but sub-optimal) solutions can be achieved in some cases by using specific methods. In Chapter 4 we have used local modifications of the global optimization problem and merging mechanisms to construct feasible solutions. In Chapter 5 we have enforced the feasibility of a subset of the constraints into the iterations of a fast iterative method. Among the other possible methods we could cite stochastic incremental subgradient methods (Johansson et al., 2009) and constraint tightening techniques (Doan et al., 2011).

With the exception of our method presented in Chapter 4 (that is non-iterative but its applicability is limited to the specified optimization problems 4.1-4.2), all these more general algorithms are still iterative (although the iterates are feasible) and an acceptable sub-optimal solution might be achieved only after several iterations. As a result these approaches might not be implementable in realistic applications, when fast and real-time solutions are required.

We believe that the development of a general approach to fast and real-time feasible distributed solutions with an acceptable level of sub-optimality for constrained convex optimization problems is of fundamental importance for the implementation of control and estimation schemes in realistic robotic networks.

Fusion of the parallel and distributed computation paradigms

In Chapter 3, we have seen the differences between the parallel and distributed computation paradigms. The first divides the computations among different cores with the intention to arrive at the same result of a standard sequential centralized algorithm. As a result, typically, parallel schemes still involve some kind of centralization via the presence of one or more computing hubs that collect all the information coming from the different cores. The second paradigm, the distributed one, which we have employed in our proposed solution (Algorithm 3.1), allows for some degree of “sub-optimality” in computing the solution and restricts the communication exchange to occur only among neighboring cores. In this way, no central data collection is needed, although the solution is typically less accurate than a standard sequential centralized algorithm. Nonetheless, in the case of Particle Filters, the final accuracy of the estimation scheme depends on the total number of particles *and* on the achieved sampling frequency, and this favors distributed schemes compared to parallel ones. We believe that the investigation of similarities and differences between parallel and distributed paradigms, and eventually their fusion in a single methodology, is essential to implement more efficient, accurate, and real-time algorithms.

Asynchronous algorithms

In all the chapters of this thesis, we have worked under the assumption that the robots (or sensors) were synchronized among each other. In this way, all the computations could happen at the same time in the whole network. Although in principle, one could synchronize the clocks of the robots, and one could do so even in a distributed way, (Simeone et al., 2008, Schenato and Fiorentin, 2011), in many realistic applications, especially for large-scale systems, this synchronization would be rather time demanding. Furthermore, if the clocks drift from each other considerably, the synchronization routine has to be kept running in the background.

Methods that could address the asynchronism of the network clocks have been studied in different areas. For the context of this thesis we cite (Boyd et al., 2006, Fagnani and Zampieri, 2008, Song et al., 2009, Oreshkin and Coates, 2010, Hu et al., 2010, Mathews and Durrant-Whyte, 2007, Xu et al., 2008, Tsitsiklis et al., 1986, Zhong and Cassandras, 2008, Johansson et al., 2009, Ram et al., 2010) that study asynchronism in estimation, control, and optimization algorithms.

We believe that a deeper analysis of the algorithms presented in this thesis, with the intention to extend them to asynchronous settings would increase their applicability in realistic scenarios. However, we also think that this extension is not an easy task when feasibility of the optimization solution is required at each time step.

Beyond discrete-time models

Another main assumption in the thesis is that discrete-time models for the mobile agents, or for the process to estimate, are available. This is often the case for mobile robots that are controlled with digital control (as in the case of an unicycle robot with discrete velocity control described in Chapter 2, or the robotic arm controlled in Chapter 3) and not subjected to any external forces. In these robotic scenarios, the methods presented in this thesis are more than reasonable and can be applied reliably on the mentioned discrete-time models.

However, in many real cases, the robots are immersed in external continuous force fields (as it is the case for autonomous underwater vehicles or satellites), or the process we would like to observe is in continuous-time. In these cases, the explicit derivation of a (nonlinear) discrete-time model could be rather demanding. Current research directions, in the field of event-triggered and quantized control and estimation, could overcome this difficulty and offer a way to design the distributed algorithm without the need of a discrete-time model. Examples of such approaches can be found in the works of (Wang and Lemmon, 2009, 2011, Mazo Jr. and Tabuada, 2011, Dimarogonas et al., 2012, De Persis and Frasca, 2012, De Persis and Bjayawardhana, 2013).

These rather new fields will be very important for designing more reliable algorithms specifically tailored to the real behavior of the underlying communication network.

Bibliography

F. Albertini and D. D'Alessandro. Remarks on the Observability of Nonlinear Discrete Time Systems. In *Proceedings of the 17th IFIP TC7 Conference on System Modelling and Optimization*, pages 155 – 162, Prague, Czech Repubblic, September 1995.

A. Alessandri, M. Baglietto, G. Battistelli, and M. Gaggero. Moving-Horizon State Estimation for Nonlinear Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, 22(5):768 – 780, 2011.

D. A. Anisi, X. Hu, and P. Ögren. Connectivity Constrained Multi-UGV Surveillance. In *Proceedings of the 17th IFAC World Congress*, pages 581 – 586, Seoul, Korea, July 2008.

Th. Arampatzis, J. Lygeros, and S. Manesis. A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In *Proceedings of the Mediterranean Conference on Control and Automation*, pages 719 – 724, Limassol, Cypros, June 2005.

K.J. Arrow and L. Hurwicz. *Essays in Economics and Econometrics*, chapter Decentralization and Computation in Resources Allocation, pages 34 – 104. University of North Carolina Press, 1960.

A. Arsie and E. Frazzoli. Efficient Routing of Multiple Vehicles with no Communications. *International Journal of Robust and Nonlinear Control*, 18(2):154 – 164, 2007.

M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174 – 188, 2002.

B. Balasingam, M. Bolic, P. M. Djuric, and J. Míguez. Efficient Distributed Resampling for Particle Filters. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3772 – 3775, Prague, Czech Republic, May 2011.

Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley Inter-Science, 2001.

A. S. Bashi, V. P. Jilkov, X. R. Li, and H. Chen. Distributed Implementations of Particle Filters. In *Proceedings of the IEEE Conference of Information Fusion*, pages 1164 – 1171, Cairns, Australia, July 2003.

D. P. Bertsekas. *Constrained Optimization and Lagrangian Multiplier Methods*. Accademic Press, 1982.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

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

P.K. Biswas and S. Phoha. Self-Organizing Sensor Networks for Integrated Target Surveillance. *IEEE Transactions on Computers*, 55(8):1033 – 1047, 2006.

M. Bolić, P. M. Djurić, and S. Hong. Resampling Algorithms and Architectures for Distributed Particle Filters. *IEEE Transactions on Signal Processing*, 53(7):2442 – 2450, 2005.

M. Bolić, A. Athalye, S. Hong, and P. Djurić. Study of Algorithmic and Architectural Characteristics of Gaussian Particle Filters. *Journal of Signal Processing Systems*, 61(2):205 – 218, 2010.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control*. 2011. In preparation, draft available at `http://www.mpc.berkeley.edu/mpc-course-material`.

J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, available at `www.vision.caltech.edu/bouguetj/calib_doc/`, 2010.

S. Boyd. Convex Optimization of Graph Laplacian Eigenvalues. In *Proceedings of the International Congress of Mathematicians*, pages 1311 – 1319, Madrid, Spain, August 2006.

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

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics. SIAM, 1994.

S. Boyd, A. Ghosh, Ba. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory, Special issue of IEEE Transactions on Information Theory and IEEE ACM Transactions on Networking,*, 14(6):2508 – 2530, 2006.

O. Brun, V. Teuliere, and J. Garcia. Parallel Particle Filtering. *Journal of Parallel and Distributed Computing*, 62(5):1186 – 1202, 2002.

D. W. Casbeer, S. Li, R. W. Beard, and R. K. Mehra. Forest Fire Monitoring With Multiple Small UAVs. In *Proceedings of the American Control Conference*, pages 3530 – 3535, Portland, USA, June 2005.

J. Casper and R. Murphy. Human-Robot Interactions during the Robot-Assisted Urban Search and Rescue Response at the World Trade Center. *IEEE Transaction on Systems, Man, and Cybernetics–Part B: Cybernetics*, 33(6):367 – 385, 2003.

F. S. Cattivelli and A. H. Sayed. Distributed Nonlinear Kalman Filtering with Applications to Wireless Localization. In *Proceedings of the 35th IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3522 – 3525, Dallas, USA, March 2010.

N. Chaamwe, W. Liu, and H. Jiang. Seismic Monitoring in Underground Mines: A Case of Mufulira Mine in Zambia: Using Wireless Sensor Networks for Seismic Monitoring. In *Proceedings of the International Conference On Electronics and Information Engineering*, pages 310 – 314, Weihai, China, August 2010.

M.-A. Chao, C.-Y. Chu, C.-H. Chao, and A.-Y. Wu. Efficient Parallelized Particle Filter Design on CUDA. In *Proceedings of the 2010 IEEE Workshop on Signal Processing Systems*, pages 299 – 304, San Francisco, USA, October 2010.

M. Chitchian, A. Simonetto, A. S. van Amesfoort, and T. Keviczky. Distributed Computation Particle Filters on GPU-Architectures for Real-Time Control Applications. *Submitted to IEEE Transactions on Control Systems Technology*, 2012a.

M. Chitchian, A. S. van Amesfoort, A. Simonetto, T. Keviczky, and H. J. Sips. Particle Filters on Multi-Core Processors. Technical report, Delft University of Technology, 2012b. Available at: `www.pds.ewi.tudelft.nl/~afoort/publ/PDS-TR2012-partfilt.pdf`.

T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray. On a Decentralized Active Sensing Strategy using Mobile Sensor Platforms in a Network. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 1914 – 1919, Paradise Island, the Bahamas, December 2004.

M. Coates. Distributed Particle Filters for Sensor Networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 99 – 107, Berkeley, USA, April 2004.

P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu. Experiments with Underwater Robot Localization and Tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4556 – 4561, Roma, Italy, April 2007.

P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental Wireless Sensor Networks. *Proceeding of IEEE*, 98(11):1903 – 1917, 2010.

J. Cortés. Distributed Algorithms for Reaching Consensus on General Functions. *Automatica*, 44(3):726 – 737, 2008.

J. Cortés, S. Martinez, T. Karatas, and F. Bullo. Coverage Control for Mobile Sensing Networks. *IEEE Transaction on Robotics and Automation*, 20(2):243 – 255, 2004.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, USA, 1991.

N. M. M. de Abreu. Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications*, 423(1):53 – 73, 2007.

M. C. De Gennaro and A. Jadbabaie. Decentralized Control of Connectivity for Multi-Agent Systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3628 – 3633, San Diego, USA, December 2006.

C. De Persis and B. Bjayawardhana. Coordination of Passive Systems under Quantized Measurements. *SIAM Journal of Control Optimization*, 2013. Accepted.

C. De Persis and P. Frasca. Self-Triggered Coordination with Ternary Controllers. In *Proceedings of the 3rd IFAC Workshop on Estimation and Control of Networked Systems*, Santa Barbara, USA, September 2012.

J. Derenick, J.R. Spletzer, and A. Hsieh. An Optimal Approach to Collaborative Target Tracking with Performance Guarantees. *Journal of Intelligent Robotic Systems*, 56(1):47 – 67, 2009.

J. Derenick, J. Spletzer, and V. Kumar. A Semidefinite Programming Framework for Controlling Multi-robot Systems in Dynamic Environments. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 7172 – 7177, Atlanta, USA, December 2010.

O. Devolder, F. Glineur, and Yu. Nesterov. A Double Smoothing Technique for Constrained Convex Optimization Problems and Applications to Optimal Control. *Submitted to SIAM Journal on Optimization*, 2011.

D.V. Dimarogonas and K.J. Kyriakopoulos. Inverse Agreement Protocols With Application to Distributed Multi-Agent Dispersion. *IEEE Transactions on Automatic Control*, 54(3):657 – 663, 2009.

D.V. Dimarogonas, E. Frazzoli, and K.H. Johansson. Distributed Event-Triggered Control for Multi Agent Systems. *IEEE Transactions on Automatic Control*, 57(5):1291 – 1297, 2012.

M. D. Doan, T. Keviczky, and B. De Schutter. A Distributed Optimization-Based Approach for Hierarchical MPC of Large-Scale Systems with Coupled Dynamics and Constraints. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5236 – 5241, Orlando, USA, December 2011.

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

F. Fagnani and S. Zampieri. Randomized Consensus Algorithms over Large Scale Networks. *IEEE Journal on Selected Areas in Communications*, 26(4):634 – 649, 2008.

S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis. Set-Membership Constrained Particle Filter: Distributed Adaptation for Sensor Networks. *IEEE Transactions on Signal Processing*, 59(9):4122 – 4138, 2011.

M. Farina, G. Ferrari-Trecate, and R. Scattolini. Distributed Moving Horizon Estimation for Linear Constrained Systems. *IEEE Transactions on Automatic Control*, 55(11):2462 – 2475, 2010.

E. Ghadimi, M. Johansson, and I. Shames. Accelerated Gradient Methods for Networked Optimization. In *Proceedings of the American Control Conference*, pages 1668 – 1673, San Francisco, USA, July 2011.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *IEE-Proceedings-F*, 140(2):107 – 113, 1993.

F. Göring, C. Helmberg, and M. Wappler. Embedded in the Shadow of the Separator. *SIAM Journal on Optimization*, 19(1):472 – 501, 2008.

B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-Theoretic Coordinated Control of Multiple Sensor Platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1521 – 1526, Taipei, Taiwan, September 2003.

D. Gu. Distributed Particle Filter for Target Tracking. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3856 – 3861, Roma, Italy, April 2007.

D. Gu and H. Hu. Target Tracking by Using Particle Filter in Sensor Networks. *International Journal of Robotics and Automation*, 24(3), 2009.

D. Gu, J. Sun, Z. Hu, and H. Li. Consensus Based Distributed Particle Filter in Sensor Network. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 302 – 307, Zhangjiajie, China, June 2008.

E.L. Haseltine and J.B. Rawlings. Critical Evaluation of Extended Kalman Filtering and Moving-Horizon Estimation. *Industrial and Engineering Chemistry Research*, 44(8):2451 – 2460, 2005.

G. Hendeby, R. Karlsson, and F. Gustafsson. Particle Filtering: The Need for Speed. *EURASIP Journal on Advances in Signal Processing*, 1(5):1 – 9, 2010.

A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: a Distributed, Scalable Solution to the Area Coverage Problem. In *Proceedings of the 6th International Symposium on Distributed an Autonomous Systems*, pages 299 – 208, Fukuoka, Japan, June 2002.

Y. Hu, Z. Duan, and D. Zhou. Estimation Fusion with General Asynchronous Multi-Rate Sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 46(4):2090 – 2101, 2010.

I. I. Hussein and D. Stipanovic. Effective Coverage Control for Mobile Sensor Networks with Guaranteed Collision Avoidance. *IEEE Transactions on Control System Technology*, 15(4):624 – 657, 2007.

S. Hutchinson, G.D. Hager, and P.I. Corke. A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation*, 12(5):651 – 670, 1996.

D. Izzo and L. Pettazzi. Autonomous and Distributed Motion Planning for Satellite Swarm. *Journal of Guidance, Control and Dynamics*, 30(2):449 – 459, 2007.

A. Jadbabaie, A. Ozdaglar, and M. Zargham. A Distributed Newton Method for Network Optimization. In *Proceedings of the 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pages 2736 – 2741, Shanghai, China, December 2009.

B. Johansson. *On Distributed Optimization in Networked Systems*. PhD thesis, KTH, Stockholm, Sweden, 2008.

B. Johansson, M. Rabi, and M. Johansson. A Randomized Incremental Subgradient Method for Distributed Optimization in Networked Systems. *SIAM Journal on Optimization*, 20(3):1157 – 1170, 2009.

S. J. Julier and J. K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of IEEE*, 93(3):401 – 422, 2004.

W. Kang. Moving Horizon Numerical Observers of Nonlinear Control Systems. *IEEE Transactions on Automatic Control*, 51(2):344 – 350, 2006.

D. Kempe and F. McSherry. A Decentralized Algorithm for Spectral Analysis. *Journal of Computer and System Sciences*, 74(1):70 – 83, 2008.

T. Keviczky and K. H. Johansson. A Study on Distributed Model Predictive Consensus. In *Proceedings of 17th IFAC World Congress*, pages 1516 – 1521, Seoul, Korea, July 2008.

Y. Kim and M. Mesbahi. On Maximizing the Second Smallest Eigenvalue of a State-Dependent Graph Laplacian. *IEEE Transactions of Automatic Control*, 51(1):116 – 120, 2006.

K.C. Kiwiel. Convergence of Approximate and Incremental Subgradient Methods for Convex Optimization. *SIAM Journal of Optimization*, 14(3):807 – 840, 2004.

J. Koshal, A. Nedić, and U. Y. Shanbhag. Multiuser Optimization: Distributed Algorithms and Error Analysis. *SIAM Journal on Optimization*, 21(3):1046 – 1081, 2011.

H. Y. K. Lau and A. W. Y. Ko. Coordination of Cooperative Search and Rescue Robots for Disaster Relief. In *Proceedings of 17th IFAC World Congress*, pages 895 – 900, Seoul, Korea, July 2008.

S. Lee and M. West. Markov Chain Distributed Particle Filters (MCDPF). In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 5496 – 5501, Shanghai, China, December 2009.

N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang. Coordinated Control of an Underwater Glider Fleet in an Adaptive Ocean Sampling Field Experiment in Monterey Bay. *Journal of Field Robotics*, 27(6):718 – 740, 2010.

N.E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D.M. Fratantoni, and R.E. Davis. Collective Motion, Sensor Networks, and Ocean Sampling. *Proceeding of IEEE*, 1(1):48 – 74, 2007.

J.-S. Li and Y.-L. Pan. de Caen's Inequality and Bounds on the Largest Laplacian Eigenvalue of a Graph. *Linear Algebra and its Applications*, 328(3):153 – 160, 2001.

H. Liu, H. So, F. Chan, and K. Lui. Distributed Particle Filtering for Target Tracking in Sensor Networks. *Progress In Electromagnetics Research C*, 11(1):171 – 182, 2009.

J. Liu, M. Chu, and J.E. Reich. Multitarget Tracking in Distributed Sensor Networks. *IEEE Signal Processing Magazine*, 24(3):36 – 46, 2007.

R. Mahadevan and F.J. Doyle III. A Partial Flatness Approach to Nonlinear Moving Horizon Estimation. In *Proceedings of the American Control Conference*, pages 211 – 215, Boston, USA, June – July 2004.

S. Martínez and F. Bullo. Optimal Sensor Placement and Motion Coordination for Target Tracking. *Automatica*, 42(4):661 – 668, 2006.

G. Mathews and H. Durrant-Whyte. Decentralised Optimal Control for Reconnaissance. In *Proceedings of the Conference on Information, Decision and Control*, pages 314 – 319, Adelaide, Australia, February 2007.

M. Mazo Jr. and P. Tabuada. Decentralized Event-Triggered Control over Sensor/Actuator Networks. *IEEE Transactions on Automatic Control*, 56(10):2456 – 2461, 2011.

J. Míguez. Analysis of Parallelizable Resampling Algorithms for Particle Filtering. *Signal Processing*, 87(12): 3155 – 3174, 2007.

A. Nedić and A. Ozdaglar. Subgradient Methods for Saddle-Point Problems. *Journal of Optimization Theory and Applications*, 142(1):205 – 228, 2009.

A. Nedić, A. Ozdaglar, and P.A. Parrilo. Constrained Consensus and Optimization in Multi-Agent Networks. *IEEE Transactions on Automatic Control*, 55(4):922 – 938, 2010.

H. Nijmeijer. Observability of Autonomous Discrete Time Non-Linear Systems: a Geometric Approach. *Internation Journal of Control*, 36(5):867 – 874, 1982.

NVIDIA. *CUDA C Programming Guide*. NVIDIA Corporation, Santa Clara, CA, USA, October 2010.

K. Oguni, T. Miyazaki, M. Saeki, and N. Yurimoto. Wireless Sensor Network for Post-seismic Building-wise Damage Detection. In *Proceedings of the 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 238 – 243, Paris, France, June 2011.

R. Olfati-Saber. Distributed Kalman Filtering for Sensor Networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 5492 – 5498, New Orleans, USA, December 2007a.

R. Olfati-Saber. Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility. In *Proceedings of the American Control Conference*, pages 4606 – 4612, New York City, USA, July 2007b.

R. Olfati-Saber and R.M. Murray. Consensus Problems in Networks of Agents with Switching Topology and Time-Delays. *IEEE Transactions on Automatic Control*, 49(9):1520 – 1533, 2004.

R. Olfati-Saber, J. A. Fax, and R.M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215 – 233, 2007.

B. N. Oreshkin and M. J. Coates. Asynchronous Distributed Particle Filter via Decentralized Evaluation of Gaussian Product. In *Proceedings of the IEEE International Conference on Information Fusion*, pages 1 – 8, Edinburgh, U.K., July 2010.

D.P. Palomar and M. Chiang. A Tutorial on Decomposition Methods for Network Utility Maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439 – 1451, 2006.

K. Par and O. Tosun. Parallelization of Particle Filter Based Localization and Map Matching Algorithms on Multicore/Manycore Architectures. In *Proceedings of the IEEE 2011 Intelligent Vehicles Symposium*, pages 820 – 826, Baden-Baden, Germany, June 2011.

C. C. Qu and J. Hahn. Computation of Arrival Cost for Moving Horizon Estimation via Unscented Kalman Filtering. *Journal of Process Control*, 19(2):358 – 363, 2009.

M. R. Rajamani and J. B. Rawlings. Improved State Estimation using a Combination of Moving Horizon Estimator and Particle Filters. In *Proceedings of the American Control Conference*, pages 4443 – 4444, New York City, USA, July 2007.

S. S. Ram, A. Nedić, and V .V. Veeravalli. Distributed Stochastic Subgradient Projection Algorithms for Convex Optimization. *Journal of Optimization Theory and Applications*, 147(3):516 – 545, 2010.

C.V. Rao. *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete Time Systems*. PhD thesis, University of Wisconsin-Madison, Madison, USA, 2000.

C.V. Rao, J.B. Rawlings, and D.Q. Mayne. Constrained State Estimation for Nonlinear Discrete-Time Systems: Stability and Moving Horizon Approximations. *IEEE Transactions on Automatic Control*, 48(2):246 – 257, 2003.

T.D. Räty. Survey on Contemporary Remote Surveillance Systems for Public Safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(5):493 – 515, 2010.

J.B. Rawlings and B.R. Bakshi. Particle Filtering and Moving Horizon Estimation. *Computers and Chemical Engineering*, 30(10):1529 – 1541, 2006.

W. Ren and R.W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*. Springer-Verlag London, 2008.

O. Rosén, A. Medvedev, and M. Ekman. Speedup and Tracking Accuracy Evaluation of Parallel Particle Filter Algorithms Implemented on a Multicore Architecture. In *Proceedings of the 2010 IEEE International Conference on Control Applications*, pages 440 – 445, Yokohama, Japan, Sept 2010.

M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized Sensor Fusion with Distributed Particle Filters. In *Proceedings of the Conference of Uncertainty in Artificial Intelligence*, pages 493 – 500, Acapulco, Mexico, August 2003.

L. Schenato and F. Fiorentin. Average TimeSynch: a Consensus-Based Protocol for Time Synchronization in Wireless Sensor Networks. *Automatica*, 47(9):1878 – 1886, 2011.

I. Shames, S. Dasgupta, B. Fidan, and B. Anderson. Circumnavigation Using Distance Measurements Under Slow Drift. *IEEE Transactions on Automatic Control*, 57(4):889 – 903, 2012.

X. Sheng and Y. Hu. Distributed Particle Filters for Wireless Sensor Network Target Tracking. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 845 – 848, Philadelphia, USA, March 2005.

X. Sheng, Y. Hu, and P. Ramanathan. Distributed Particle Filter with GMM Approximation for Multiple Targets Localization and Tracking in Wireless Sensor Network. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 181 – 188, Los Angeles, USA, April 2005.

O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. Strogatz. Distributed Synchronization in Wireless Networks. *IEEE Signal Processing Magazine*, 25(5):81 – 97, 2008.

A. Simonetto and T. Keviczky. Recent Developments in Distributed Particle Filters: Towards Fast and Accurate Algorithms. In *Proceedings of the 1st IFAC Workshop on Estimation and Control of Networked Systems*, pages 138 – 143, Venice, Italy, September 2009.

A. Simonetto and T. Keviczky. Distributed Multi-Target Tracking via Mobile Robotic Networks: a Localized Non-iterative SDP Approach. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 4226 – 4231, Orlando, USA, December 2011.

A. Simonetto and T. Keviczky. *Distributed Decision Making and Control*, volume 417 of *Lecture Notes in Control and Information Sciences*, chapter Distributed Nonlinear Estimation for Diverse Sensor Devices, pages 147 – 169. Springer, 2012.

A. Simonetto, T. Keviczky, and R. Babuška. Distributed Nonlinear State Estimation of Mobile Robots via Sensor Networks . In preparation: to be submitted as Springer Brief, 2012.

A. Simonetto, P. Scerri, and K. Sycara. A Mobile Network for Mobile Sensors. In *Proceedings of the IEEE International Conference on Information Fusion*, pages 1 – 8, Cologne, Germany, June – July 2008.

A. Simonetto, T. Keviczky, and R. Babuška. Distributed Nonlinear Estimation for Robot Localization using Weighted Consensus. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3026 – 3031, Anchorage, USA, May 2010a.

A. Simonetto, T. Keviczky, and R. Babuška. Distributed Algebraic Connectivity Maximization for Robotic Networks: A Heuristic Approach. In *Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems*, Lausanne, Switzerland, November 2010b.

A. Simonetto, D. Balzaretti, and T. Keviczky. Evaluation of a Distributed Moving Horizion Estimator for a Mobile Robot Localization Problem. In *Proceedings of the 18th IFAC World Congress*, pages 8902 – 8907, Milan, Italy, August – September 2011a.

A. Simonetto, T. Keviczky, and R. Babuška. On Distributed Algebraic Connectivity Maximization in Robotic Networks. In *Proceedings of the American Control Conference*, pages 2180 – 2185, San Francisco, USA, June – July 2011b.

A. Simonetto, T. Keviczky, and R. Babuška. Constrained Distributed Algebraic Connectivity Maximization in Robotic Networks. *Submitted to Automatica*, 2012a.

A. Simonetto, T. Keviczky, and D.V. Dimarogonas. Distributed Solution for a Maximum Variance Unfolding Problem with Sensor and Robotic Network Applications. 2012b. *Presented at the 50th Allerton Conference*, Allerton, USA, October 2012.

A. Simonetto, T. Keviczky, and M. Johansson. A Regularized Saddle-Point Algorithm for Networked Optimization with Resource Allocation Constraints. 2012c. *To be presented at the 51st IEEE Conference on Decision and Control*, Maui, USA, December 2012.

A. Simonetto, T. Keviczky, and R. Babuška. *Distributed Autonomous Robotic Systems*, volume 83 of *STAR*, chapter Distributed Algebraic Connectivity Maximization for Robotic Networks: A Heuristic Approach, pages 267 – 279. Spriger, 2013.

A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *Proceedings of MobiSYS*, pages 190 – 202, Boston, USA, June 2004.

C. Song, H. Zhao, and W. Jing. Asynchronous Distributed PF Algorithm for WSN Target Tracking. In *Proceedings of the International Conference on Wireless Communication and Mobile Computing*, pages 1168 – 1172, Leipzig, Germany, June 2009.

Oh Songhwai, L. Schenato, P. Chen, and S. Sastry. Tracking and Coordination of Multiple Agents Using Sensor Networks: System Design, Algorithms and Experiments. *Proceedings of the IEEE*, 95(1):234 – 254, 2007.

J. R. Spletzer and C. J. Taylor. Dynamic Sensor Planning and Control for Optimally Tracking Targets. *International Journal of Robotic Research*, 22(1):7 – 20, 2003.

G. Sun, G. Qiao, and B. Xu. Corrosion Monitoring Sensor Networks with Energy Harvesting. *IEEE Sensors Journal*, 11(6):1476 – 1477, 2011.

J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The Fastest Mixing Markov Process on a Graph and a Connection to a Maximum Variance Unfolding Problem. *SIAM Review*, 48(4):681 – 699, 2006.

T. Sun, Ling-Jyh Chen, Chih-Chieh Han, and M. Gerla. Reliable Sensor Networks for Planet Exploration. In *Proceedings of the IEEE Networking, Sensing and Control conference*, pages 816 – 821, Tucson, USA, March 2005.

V. A. Susan and S. Dubowsky. Visually Guided Cooperative Robot Actions Based on Information Quality. *Autonomous Robots*, 19:89 – 110, 2005.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

J.N. Tsitsiklis, D.P. Bertsekas, and M. Athans. Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms. *IEEE Transaction on Automatic Control*, 31(9):803 – 812, 1986.

S. Ungarala. Computing Arrival Cost Parameters in Moving Horizon Estimation Using Sampling Based Filters. *Journal of Process Control*, 19(9):1576 – 1588, 2009.

D. van der Lijn, G.A.D. Lopes, and R. Babuska. Motion Estimation Based on Predator/Prey Vision. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3435 – 3440, Taipei, Taiwan, October 2010.

X. Wang and M. Lemmon. Self Triggered Feedback Control Systems with Finite Gain L2 Stability. *IEEE Transactions on Automatic Control*, 54(3):452 – 467, 2009.

X. Wang and M. Lemmon. Event-Triggered in Distributed Networked Control Systems. *IEEE Transactions on Automatic Control*, 56(3):586 – 601, 2011.

K. Q. Weinberger and L. K. Saul. Unsupervised Learning of Image Manifolds by Semidefinite Programming. *International Journal of Computer Vision*, 70(1):77 – 90, 2006.

K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul. *Advances in Neural Information Processing Systems 19*, chapter Graph Laplacian Regularization for Large-Scale Semidefinite Programming, pages 1489 – 1496. MIT Press, Cambridge, MA, 2007.

L. Xiao and S. Boyd. Optimal Scaling of a Gradient Method for Distributed Resource Allocation. *Journal of Optimization Theory and Applications*, 129(3):469 – 488, 2006.

L. Xiao, S. Boyd, and S. Lall. A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 63 – 70, Los Angeles, USA, April 2005.

L. Xiao, S. Boyd, and S. Lall. A Space-Time Diffusion Scheme for Peer-to-Peer Least-Squares Estimation. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 168 – 176, Nashville, USA, April 2006.

Y. Xu, P. Scerri, M. Lewis, and K. Sycara. *Cooperative Networks: Control and Optimization*, chapter 1. Token-Based Approach for Scalable Team Coordination. Edward Elgar, 2008.

M. Zhong and C. G. Cassandras. Asynchronous Distributed Optimization with Minimal Communication. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 363 – 368, Cancun, Mexico, December 2008.

K. Zhou and S.I. Roumeliotis. Optimal Motion Strategies for Range-Only Constrained Multisensor Target Tracking. *IEEE Transactions on Robotics*, 24(5):1168 – 1185, 2008.

M. Zhu and S. Martínez. On Distributed Convex Optimization Under Inequality and Equality Constraints. *IEEE Transactions on Automatic Control*, 57(1):151 – 164, 2012.

Symbols and Abbreviations

General Notation

Fields	
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_0$	Set of non-zero real numbers
$\mathbb{R}_+$	Set of non-negative real numbers

Constants and Indices	
i, j, p, q	Indices indicating different devices and targets
k, τ	Discrete time indices
N	Number of mobile or non-moving devices
I_n	Identity matrix of dimension $n \times n$
0_n	Null matrix of dimension $n \times n$
$\mathbf{1}_n$	Column vector of dimension n with all entries 1
$\mathbf{0}_n$	Column vector of dimension n with all entries 0

Operations on Vectors, Matrices, and functions	
$\lvert\lvert a\rvert\rvert$	Euclidean norm
$\lvert\lvert a\rvert\rvert_A^2$	$= a^\top Aa$, where a is a vector and A a positive definite matrix of appropriate dimensions
$\langle a, b\rangle$	$= a^\top b$, where a and b are vectors of appropriate dimension
$\nabla_a f(a)$	(Sub)gradient of f with respect to a
$a_i(k)$	Value of the variable a for the device i at the discrete time k
$\delta a(k)$	$= a(k) - a(k-1)$
$\otimes$	Kroenecker product
$\mathbb{E}[a]$	Expected value of a
$\aleph(a, A)$	Gaussian distribution with mean a and covariance A

Graphs	
$\mathcal{V}$	Node set
$\mathcal{E}$	Edge set

$\mathcal{G}$	Graph, i.e., $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
L	Laplacian of the graph
ℓ_{ij}	Element ij of the Laplacian of the graph
λ_2	Algebraic connectivity
$\lambda_{\max}$	Maximum eigenvalue of the Laplacian matrix
$\mathcal{N}_i$	Neighbors set of agent i
$\mathcal{N}_i^+$	$\mathcal{N}_i \cup \{i\}$

Dynamical Systems

$\mathbf{x}$	State vector
x	Position vector
v	Velocity vector
Δt	Sampling time

Chapter 2

Symbols in order of appearance

$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{w}(k))$	Nonlinear dynamical model
$\mathbf{w}$	Process noise
$\mathbb{X}, \mathbb{W}, \mathbb{M}_i$	Constraint sets
$\mathbf{z}_i$	Local measurements
$\mathbf{z}_i(k) = g_i(\mathbf{x}(k)) + \boldsymbol{\mu}_i(k)$	Nonlinear measurement equation
$\bar{\mathbf{x}}_i, \bar{P}_i$	Local estimate and covariance before agreement
$\hat{\mathbf{x}}_i, \hat{P}_i$	Local estimate and covariance after agreement
τ	Iteration number for the agreement process
W	Consensus matrix
R_i, Q	Weight matrices
$\mathbf{x}^{\text{in}}(0), P^{\text{in}}(0)$	Initial estimate and covariance
$J_k, \hat{J}_k, \hat{\Phi}_k$	(Approximated) cost functions
$Z_{k-T}, \hat{Z}_{k-T}$	Real and approximated arrival cost
$\beta(k-T)$	Scaling factor
$\hat{\mathbf{x}}^{\text{mh}}(k-T), \hat{P}^{\text{mh}}(k-T)$	Solution at the beginning of the moving window
$\pi_{\mathbf{w}}(\mathbf{w}), \pi_{\boldsymbol{\mu}_i}(\boldsymbol{\mu}_i)$	PDF that model process and measurement noise
$(\mathbf{x}(k-1)^j, \mathbf{w}(k-1)^j)$	Particle-weight couple
$p(\mathbf{x}(k)\|\mathbf{z}(k))$	A posteriori distribution
$q(\mathbf{x}(k)\|\mathbf{z}(k))$	Proposal distribution

Abbreviations

MHE	Moving Horizon Estimator
PF	Particle Filter
UKF	Unscented Kalman Filter
EKF	Extended Kalman Filter

Chapter 3

Symbols in order of appearance	
N	Number of local filters
m	Number of particles on each computing unit
t	Exchanged particles
$\hat{p}_i(\mathbf{x}(k)\|\mathbf{z}(k))$	Local approximation of the a posteriori PDF
$\hat{p}_i^{(t)}(\mathbf{x}(k)\|\mathbf{z}(k))$	Local approximation of the a posteriori PDF using t particles
$D(p_1, p_2)$	KullbackLeibler (KL) divergence between the distributions p_1 and p_2

Chapter 4

Symbols in order of appearance	
w	Weights for the edges of the Laplacian
d_{ij}^2	Square distance between agent i and j
f_d, f_w	Distance and weighting functions
ρ_1, ρ_2	Weighting function's parameters
γ	Optimization variable
c^d, c^w	Partial derivatives of f_d and f_w
$\triangle$	Linearizing operator, i.e., $\triangle f(k) = f(k-1) + \nabla f \delta f(k)$
$\triangle\mathcal{Q}_1, \triangle\mathcal{Q}_2$	Sets of constraints
$\mathcal{S}_{\triangle\mathcal{Q}_2}$	Parameter set
u_i	Control input of agent i
A_{1i}, A_{2i}, b_{1i}	(Block) elements of the matrices of the dynamical system of agent i
$\bar{\mathcal{U}}_i$	Closed polytopic set in which u_i is constrained
H_i, h_i	Matrix and vector that describe the set $\bar{\mathcal{U}}_i$
$\mathbf{u}_i(k)$	Lifted control input for agent i
$\mathbf{x}_i(k+1) = \mathcal{D}_i(\mathbf{x}_i(k), \mathbf{u}_i(k))$	Short-hand notation for the discrete-time LTI dynamical system of agent i as used in the optimization problem
$\mathcal{U}_i$	$\bar{\mathcal{U}}_i \times \bar{\mathcal{U}}_i$
$\mathcal{F}_i, \mathcal{F}_{x,i}, \mathcal{F}_{v,i}$	Invariant sets for the optimization problem
$\mathcal{J}_i$	Enlarged neighborhood set of cardinality J_i
n_i	Size of the enlarged neighborhood set
$L_{i,n_i}, \mathcal{E}_{i,n_i}, \mathbf{x}_{\mathcal{J}_i}, \mathbf{u}_{\mathcal{J}_i}$	Variables referring to the enlarged neighborhood set
$\tilde{a}$	Local version of the variable, set, or dynamical system a
γ_i	Local optimization variable
$\mathcal{Q}_3$	Set of constraints
$\tilde{\mathbf{x}}_{ij}, \tilde{\mathbf{u}}_{ij}$	State or control of the agent j as computed by the agent i
$\mathcal{J}_i^*$	Set: $\{p\|i \in \mathcal{J}_p\}$
s_i	Constant of a positive linear combination
$\bar{s}$	$\sum_{i=1}^{N} s_i$, assumed to be ≤ 1

$\bar{s}_{ij}$	$\sum_{p\in\mathcal{J}_i^*\cap\mathcal{J}_j^*} s_p$
$\tilde{x}^{(i)}$	Local solutions defined in (4.45)
$\tilde{x}^{(i)}\|_{n_i}$, $\tilde{x}\|_{\mathbf{n}}$	Local solutions defined in Definition 4.2 and 4.3
e_i^+, e_i^-	Local sub-optimality measures
M	Number of targets
z	Position vector of the targets
z^*	Worst case position vector of the targets
$\mathbf{w}_q$	Input associated to target q
$v_{\max,i}$	Maximal velocity for agent i
$\mathbf{w}_{\max,q}$	Maximal velocity associated to target q
$\mathcal{Z}_q(k)$	Set of reachable positions at time k for target q
$\mathcal{R}_q$	Set that collects the agents that see target q
f_V	Visual weight function
v	Visual weight
$\mathrm{supp}(f)$	Support of the function f
$\alpha \geq 0, \beta_q \geq 0$	Constant relative weights in the optimization problems
ν_q	Optimization variable
c^{v}	Partial derivative of the visual weighting function
ν_q^-	Decrease of detection quality due to targets' motion, Definition 4.4
$\mathcal{T}_i$	Set of all the targets that agent i is aware of
$\mathcal{O}_q$	Enlarged neighborhood set for target q
$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$	Standard form for a discrete-time LTI dynamical system

Chapter 5

Symbols in order of appearance

x_i	Nodal optimization variable
f_i, g_{ij}, h_i	Non-strictly convex functions
x_{tot}	Total resource vector
$\bar{\mathbb{X}}, \mathbb{X}, \hat{\mathbb{X}}, \hat{\mathbb{M}}$	Convex constraint sets
$\mathcal{L}$	Lagrangian function
ν, ϵ	Regularization parameters
μ	Dual variable of x
a^{opt}	Optimal value of the variable a
a^*	Optimal value of the variable a in the regularized problem
τ	Iteration counter
$a^{(\tau)}$	Value of the variable a at the iteration τ
$\mathcal{P}_{\mathbb{A}}[\cdot]$	Projection on the set $\mathbb{A}$, e.g. $\mathcal{P}_{\mathbb{R}_+}[\cdot]$
W	Weight matrix
α, β	Step-sizes
p	Lagrangian multiplier
z	Stacked vector $(x^\top, \mu^\top)^\top$

p	Lagrangian multiplier
$\Phi(z)$	Mapping $(\nabla_x \mathcal{L}(z)^\top, -\nabla_\mu \mathcal{L}(z)^\top)^\top$
$W_\otimes$	$:= W \otimes I_n$
F, G_q	Lipschitz constants of the gradients of f and g_q
M_d, M_μ	Bounds on the gradient of g and on the dual variable μ
φ	Strong monotonicity constant of the function Φ
F_Φ	Lipschitz constant of the function Φ
C	$:= \max(\beta\lambda_{\max}(W), 1)$
κ	$:= \varphi - F_\Phi\,(\beta\lambda_{\max}(W) - 1)$
y	Position of the target
r_{ij}	Bound associated with the edge (i, j)
X	Matrix optimization variable
w_{ij}	Weight for the edge i, j
c	Cost of the MVU problem
$\mathbf{v}_2$	Eigenvector associated to λ_2
$\mathbf{q_{ij}}$	Vector defined in (5.39)
$\Lambda, \Lambda_\varepsilon, \bar{\Lambda}_\varepsilon^2$	Bounds on the maximal value of the subgradient
$\bar{a}^{(\tau)}$	Running average of $a^{(\tau)}$
$\nabla_{a,\varepsilon} f(a)$	ε-subgradient of f with respect to a
ε	Level of error in the subgradient calculations

Summary

Distributed Estimation and Control for Robotic Networks

Andrea Simonetto

Mobile robots that communicate and cooperate to achieve a common task have been the subject of an increasing research interest in recent years. These possibly heterogeneous groups of robots communicate locally via a communication network and therefore are usually referred to as *robotic networks*. Their potential applications are diverse and encompass monitoring, exploration, search and rescue, and disaster relief. From a research standpoint, in this thesis we consider specific aspects related to the foundations of robotic network algorithmic development: distributed estimation, control, and optimization.

The word “distributed” refers to situations in which the cooperating robots have a limited, local knowledge of the environment and of the group, as opposed to a “centralized” scenario, where all the robots have access to the complete information. The typical challenge in distributed systems is to achieve similar results (in terms of performance of the estimation, control, or optimization task) with respect to a centralized system without extensive communication among the cooperating robots.

In this thesis we develop effective distributed estimation, control, and optimization algorithms tailored to the distributed nature of robotic networks. These algorithms strive for limiting the local communication among the mobile robots, in order to be applicable in practical situations. In particular, we focus on issues related to nonlinearities of the dynamical model of the robots and their sensors, to the connectivity of the communication graph through which the robots interact, and to fast feasible solutions for the common (estimation or control) objective.

First, we investigate a nonlinear state estimation problem via a stationary robotic network (often referred to as “sensor network”). Typically, in the distributed setting, only linear time-invariant processes have been considered. Motivated by the number of application scenarios where such a linear model would not be adequate, we instead present a unified way of describing distributed implementations of four commonly used nonlinear estimators: the Moving Horizon Estimator, the Particle Filter, the Extended and Unscented Kalman Filter. Making use of the unifying framework, we propose new distributed versions of these methods, in which the nonlinearities are locally managed by the various sensor nodes and the different estimates are merged, based on a weighted average consensus process. We show how the merging mechanism can handle different filtering algorithms

implemented on heterogeneous sensors, which is especially useful when they are endowed with diverse local computational capabilities.

As a second step, we focus on methods that reduce the high computational requirements of the mentioned nonlinear estimators by distributing the computations among different computing devices communicating with one another. We aim to deliver real-time implementable solutions for the nonlinear estimation task, which can be used in realistic robotic networks. In particular, we propose parallel, networked formulations of Particle Filters and investigate how they can be implemented on GPU-architectures, for real-time control or monitoring applications. We validate our approach via experimental and numerical results which support the idea that real-time nonlinear estimation is achievable by a suitable adaptation of the Particle Filter algorithm and widely available parallel computing platforms.

Then, we turn our attention to investigating distributed control problems related to the connectivity of the communication network among the robots. Typically in the available literature, this has been considered to be guaranteed and available by assumption rather than being seen as an objective to be reached via distributed control actions. In addition, the concept of "how well-connected" the robotic network is, has often been overlooked, whereas this property can significantly improve the performance (e.g., the convergence rate) of the distributed algorithms that run on the network. In this respect, we formulate a control problem leading to the maximization of the connectivity of the robotic network measured by the so-called algebraic connectivity of the network graph. We propose a non-iterative and feasible (thus implementable in practical situations) distributed solution to this problem which moves the mobile robots in more favorable positions in terms of connectivity. This distributed solution is based on a sequence of local Semi-Definite Programs (SDP) formulated using state-dependent graph Laplacians. We then proceed to extend and utilize the presented distributed method to tackle the problem of collectively tracking a number of moving targets while maintaining a certain level of connectivity among the network of mobile robots. Numerical simulations show the performance of the distributed algorithms with respect to the centralized solutions.

Finally, we focus on convex and non-convex networked optimization problems with resource allocation constraints, which can be used in realistic robotic network applications where the mobile or non-moving robots share the same resources among each other. We propose a regularized saddle-point algorithm for convex networked optimization problems with resource allocation constraints. Standard subgradient methods suffer from slow convergence and require excessive communication when applied to problems of this type. Our approach offers an alternative way to address these problems, and ensures that each iterative update step satisfies the resource allocation constraints. Then, we investigate a particular non-convex networked resource allocation problem, known as the Maximum Variance Unfolding problem and its dual, the Fastest Mixing Markov Process problem. These problems are of relevance for sensor networks and mobile robot applications. We solve both these problems with the same distributed primal-dual subgradient iterations whose convergence is proven even in the case of approximation errors in the calculation of the subgradients (which is of practical importance for generating real-time solutions). Finally, we illustrate the importance of the presented algorithms for target tracking in robotic networks, localization, and dispersion problems.

Samenvatting

Gedistribueerde Schatting en Regeling voor Robotnetwerken

Andrea Simonetto

Mobiele robots die communiceren en samenwerken om een gemeenschappelijke taak te vervullen zijn de afgelopen jaren in een toenemende wetenschappelijke belangstelling komen te staan. Deze mogelijkerwijs heterogene groepen robots communiceren lokaal via een communicatienetwerk en worden daarom vaak *robotnetwerken* genoemd. Hun potentiële toepassingen zijn divers en omvatten controle, verkenning, opsporing en redding en hulp bij calamiteiten. Vanuit een onderzoeksstandpunt beschouwen we in dit proefschrift specifieke fundamentele aspecten gerelateerd aan de ontwikkeling van algoritmes voor robotnetwerken: gedistribueerd schatten, regelen en optimaliseren.

Het woord "gedistribueerd" verwijst naar situaties waarin de samenwerkende robots beschikken over beperkte lokale kennis over hun omgeving en over de groep, dit in tegenstelling tot een "gecentraliseerd" scenario waarin alle robots volledige informatie tot hun beschikking hebben. Een kenmerkende uitdaging in gedistribueerde systemen is om vergelijkbare resultaten (in termen van de prestatie van de schattings-, regel- of optimalisatietaak) te behalen als in een gecentraliseerd system, zonder grootschalige communicatie tussen de samenwerkende robots.

In dit proefschrift ontwikkelen we effectieve gedistribueerde schattings-, regel- en optimalisatiealgoritmes toegesneden op het gedistribueerde karakter van robotnetwerken. Deze algoritmes streven naar een beperkte communicatie tussen de mobiele robots om zo toepasbaar te zijn in praktische situaties. In het bijzonder richten we ons op kwesties gerelateerd aan de niet-lineariteiten in de dynamische modellen waarmee de robots en hun sensoren beschreven worden, aan de connectiviteit van de communicatiegraaf die ten grondslag ligt aan de communicatie tussen de robots en aan snelle, haalbare oplossingen voor het gemeenschappelijke (schattings- of regel-)doel.

Allereerst onderzoeken we een niet-lineair toestandschattingsprobleem in een stationair robotnetwerk (vaak een "sensornetwerk" genoemd). In een gedistribueerd kader zijn tot op heden vaak slechts lineaire, tijdinvariante processen beschouwd. Gemotiveerd door het aantal scenario's waarin een dergelijk lineair model niet toereikend zou zijn, presenteren we een algemene manier om gedistribueerde implementaties van vier gangbare niet-lineaire schatters te verkrijgen: de *Moving Horizon* schatter, het *Particle Filter*, het *Extended Kalman Filter* en het *Unscented Kalman Filter*. Gebruik makend van dit algemene

raamwerk, doen we voorstellen voor nieuwe gesdistribueerde varianten van deze methodes waarin de niet-lineariteiten lokaal door de verschillende sensorknooppunten geregeld worden en waarbij de verschillende schattingen samengevoegd worden, gebaseerd op een consensusproces via het gewogen gemiddelde. We laten zien hoe het samenvoegingsmechanisme om kan gaan met verschillende filteralgoritmes die geïmplementeerd zijn op heterogene sensoren, wat in het bijzonder bruikbaar is in het geval de lokale rekenvermogens van deze sensoren van elkaar verschillen.

Als tweede stap richten we ons op methodes die de zware rekenkrachteisen van de genoemde niet-lineaire schatters reduceren door de berekeningen te verdelen over verschillende rekenapparaten die met elkaar communiceren. Ons doel is real-time implementeerbare oplossingen te leveren voor de niet-lineaire schattingstaak, die gebruikt kunnen worden in realistische robotnetwerken. We stellen voornamelijk parallelle, genetwerkte formuleringen voor van Particle Filters en onderzoeken hoe deze geïmplementeerd kunnen worden op GPU-architecturen voor real-time regel- of monitoringtoepassingen. We valideren onze benadering op basis van experimentele en numerieke resultaten die het idee ondersteunen dat real-time niet-lineaire schatting bereikbaar wordt door passende modificaties van het Particle Filter algoritme en goed verkrijgbare parallelle rekenplatforms.

Vervolgens richten we onze aandacht op het onderzoeken van gedistribueerde regelproblemen in relatie tot de connectiviteit van het communicatienetwerk tussen de robots onderling. In de literatuur wordt veelal verondersteld dat deze connectiviteit gegarandeerd aanwezig en bekend is, in plaats van beschouwd te worden als een doel, te bereiken via gedistribueerde regelacties. Bovendien wordt “hoe goed onderling verbonden” het robotnetwerk is vaak over het hoofd gezien, terwijl deze eigenschap de prestaties (bv. de convergentiesnelheid) van gedistribueerde algoritmes die op het netwerk draaien significant kan verbeteren. In dit kader formuleren we een regelprobleem dat leidt tot maximalisatie van de connectiviteit van het robotnetwerk, uitgedrukt in de zogenaamde algebraïsche connectiviteit van de netwerkgraaf. We stellen een niet-iteratieve en haalbare (dus implementeerbaar in praktische situaties) gedistribueerde oplossing voor dit probleem voor, die de mobiele robots doet verplaatsen naar gunstigere posities in termen van connectiviteit. Deze gedistribueerde oplossing is gebaseerd op een reeks lokale semidefiniete programma’s uitgedrukt met behulp van toestandsafhankelijke Laplacianen van de graaf. Vervolgens breiden we deze gedistribueerde methode uit en gebruiken deze om het probleem op te lossen waarbij een aantal bewegende doelen collectief gevolgd dient te worden terwijl een zekere connectiviteit tussen de mobiele robots gewaarborgd blijft. Numerieke simulaties tonen de prestaties van de gedistribueerde algoritmes aan in vergelijking met de gecentraliseerde oplossingen.

Ten slotte concentreren we ons op convexe en niet-convexe optimalisatieproblemen met beperkingen op de toewijzing van rekenkracht, die gebruikt kunnen worden in realistische robotnetwerktoepassingen waar de mobiele of vaste robots een hoeveelheid rekenkracht onderling delen. We stellen een geregulariseerd zadelpuntalgoritme voor waarmee convexe optimalisatieproblemen in netwerkverband kunnen worden opgelost met beperkingen op de toewijzing van rekenkracht. Standaard subgradiëntmethodes lijden onder trage convergentie en vereisen een grote hoeveelheid communicatie wanneer deze toegepast worden op problemen van dit type. Onze benadering biedt een alternatieve manier om deze problemen aan te pakken en garandeert dat elke iteratieve correctiestap aan de beperkingen op de toewijzing van rekenkracht voldoet. Daarna onderzoeken we een specifiek niet-convex *re-*

source allocation probleem in netwerkverband, bekend als het *Maximum Variance Unfolding* probleem en het duale probleem hiervan, het *Fast Mixing Markov Process* probleem. Deze problemen zijn relevant voor sensornetwerken en toepassingen met mobiele robots. We lossen beide problemen op met dezelfde primaire-duale subgradiëntiteraties waarvan de convergentie bewezen is, zelfs in het geval van benaderingsfouten in de berekening van de subgradiënten (hetgeen van praktisch belang is voor het genereren van real-time oplossingen). Tot besluit illustreren we het belang van de besproken algoritmes for het volgen van bewegende doelen in robotnetwerken, lokalisatie en dispersieproblemen.

Sommario

Stima e Controllo Distribuiti per Reti di Robot

Andrea Simonetto

Robot che comunicano e cooperano tra loro per realizzare una missione comune sono un soggetto di ricerca di crescente interesse. Questi gruppi di robot, talvolta di tipologie differenti, comunicano tra loro localmente su una rete di comunicazione e sono chiamati "reti di robot" per questo motivo. Le loro possibili applicazioni sono molteplici e comprendono il monitoraggio, l'esplorazione, la ricerca, il salvataggio e il soccorso in caso di disastri. Da un punto di vista scientifico, in questa tesi si sono prese in considerazione le fondazioni teoriche del progetto di algoritmi per reti di robot: stima, controllo ed ottimizzazione distribuiti.

Il termine "distribuito" denota la situazione in cui i robot hanno una conoscenza parziale e locale dell'ambiente circostante e del gruppo di robot con cui cooperano, al contrario dello scenario "centralizzato" in cui tutti i robot hanno a disposizione un'informazione completa. Nell'ambito dei sistemi distribuiti, la sfida ritenuta più comune è di ottenere risultati paragonabili a quelli un sistema centralizzato in termini di stima, controllo ed ottimizzazione senza un utilizzo eccessivo di comunicazioni tra i robot per la cooperazione.

In questa tesi si sono sviluppati algoritmi efficienti per la stima, il controllo e l'ottimizzazione. Gli algoritmi sono specificamente progettati per la natura distribuita delle reti di robot e, per essere applicabili a situazioni realistiche, limitano le comunicazioni intrarobot. In particolare, ci si è soffermati sulle problematiche riguardanti le nonlinearità del modello dinamico dei robot e dei loro sensori, la connettività del grafo di comunicazione attraverso cui i robot comunicano, e la generazione veloce di soluzioni ammissibili per i problemi analizzati.

In primo luogo, si è studiato il problema della stima di uno processo nonlineare attraverso una rete di robot stazionari (nota anche come "rete di sensori"), in quanto in ambito distribuito, solo processi lineari tempo-invarianti erano stati presi in considerazione. Motivati dal gran numero di scenari applicativi in cui tali modelli lineari non sono adeguati, si è presentato un metodo unificato per descrivere implementazioni distribuite di quattro stimatori nonlineari di comune utilizzo: lo Stimatore ad Orizzonte Mobile (*Moving Horizon Estimator*), il Filtro Particellare (*Particle Filter*) ed il Filtro di Kalman Esteso ed *Unscented*. Utilizzando tale approccio unificato, si sono introdotte nuove versioni distribuite di questi metodi, in cui le nonlinearità sono gestite localmente dai vari nodi sensore e le diverse stime sono fuse in base ad un processo di consenso a media pesata. Si è inoltre mostrato

come i meccanismi di fusione delle stime sono in grado di gestire diversi algoritmi di filtraggio implementati su sensori eterogenei. Questo è particolarmente utile quando la rete di sensori è formata da sensori con diverse capacità computazionali.

In secondo luogo, ci si è concentrati su metodi che riducono l'elevato costo computazionale degli stimatori nonlineari sopra elencati distribuendo le operazioni su processori differenti comunicanti tra loro. Lo scopo era la generazione in tempo reale di soluzioni per il problema di stima nonlineare, soluzioni che possano essere utilizzate in reti di robot realistiche. In particolare, si sono proposte versioni parallele e distribuite del Filtro Particellare, e si sono investigate le loro possibili implementazioni su architetture basate su GPU, per applicazioni di controllo e di monitoraggio in tempo reale. L'approccio è stato poi verificato attraverso studi numerici e sperimentali che hanno confermano la possibilità della stima nonlineare in tempo reale tramite un'opportuna modifica del Filtro Particellare e l'utilizzo di comuni piattaforme per il calcolo parallelo.

In seguito, ci si è interessati allo studio di problemi di controllo distribuito riguardanti la connettività della rete di comunicazione tra i robot. Nei metodi disponibili al giorno d'oggi, la connettività è spesso data per garantita o assunta come ipotesi, piuttosto che vista come un obiettivo da raggiungere attraverso azioni di controllo distribuito. Inoltre, il concetto di "quanto" una rete è connessa è spesso ignorato, mentre questa proprietà può migliorare in modo significativo le prestazioni (per esempio l'ordine di convergenza) di un algoritmo distribuito che viene eseguito da una rete. A questo riguardo, si è formulato un problema di controllo che porta alla massimizzazione della connettività di una rete di robot misurata dalla cosiddetta connettività algebrica del grafo di rete. Si è poi proposta una soluzione distribuita non iterativa e ammissibile (per questo motivo implementabile in situazioni reali) che guida i robot in una configurazione più favorevole in termini di connettività. Questa soluzione distribuita si basa su una sequenza di Problemi Semi-Definiti locali formulati usando Laplaciani di grafo dipendenti dalle posizioni dei robot. In seguito si è proceduto all'estensione e l'utilizzo del metodo distribuito precedentemente sviluppato al problema dell'inseguimento collettivo di un numero di obiettivi mobili mantenendo un certo livello di connettività tra i robot nella rete. Simulazioni numeriche hanno mostrato le prestazioni degli algoritmi distribuiti rispetto a soluzioni centralizzate.

Per finire, si sono considerati problemi di ottimizzazione convessi e non convessi su reti con limiti imposti alle risorse disponibili. Per prima cosa, si è proposto un algoritmo a punto di sella regolarizzato per problemi convessi con questo tipo di vincoli. Metodi standard basati sul sub-gradiente soffrono di convergenza lenta e necessitano di comunicazioni eccessive in questi casi. L'approccio presentato offre invece un metodo alternativo di risolvere questo problema vincolato e garantisce che ogni aggiornamento iterativo rispetti il limite sulle risorse disponibili. In seguito, si è investigato un particolare problema non-convesso di distribuzione di risorse su rete, noto come lo Spiegamento a Massima Varianza (*Maximum Variance Unfolding*) e il suo duale, il Processo di Markov a Mischiaggio più Veloce (*Fastest Mixing Markov Process*). Questi problemi sono rilevanti per applicazioni con reti di sensori e robot mobili. Si sono risolti entrambi i problemi con lo stesso metodo primale-duale distribuito basato su sub-grandienti la cui convergenza è stata dimostrata anche nel caso di errori di approssimazione nel calcolo dei sub-gradienti (il che è di importanza pratica per la generazione di soluzioni in tempo reale). Infine, si è dimostrata l'importanza degli algoritmi presentati per problemi di monitoraggio, localizzazione e dispersione in reti di robot.

Curriculum Vitae

I was born in February 1983 in Cantù, nearby Milano, Italy and I grew up in the village of Vertemate con Minoprio. During my high-school years I was awarded with an honorable mention at the 33rd International Physics Olympiad (IPhO) in Bali, Indonesia, 2002. After that, I got my B.Sc. in Aerospace Engineering with honors at Politecnico di Milano in 2005. From the end of 2005 I was enrolled in a double-degree master program between Politecnico di Milano and Torino and I earned my M.Sc. in Space Engineering with honors in April 2008. The title of the thesis was: "Distributed and Robust Control for Space Multi-Agent Systems" supervised by Dr. M. Lavagna and Dr. P. Maggiore. I was awarded with a gold medal as the best graduate in Space Engineering of the year from Politecnico di Milano. In the same period I also graduate from Alta Scuola Politecnica (December 2007) and I spent research visiting periods at European Space Agency in ESTEC, Noordwijk, the Netherlands and at the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA (under the supervision of Prof. dr. K. Sycara and Dr. P. Scerri).

Since September 2008 I have been working towards my PhD degree at the Delft Center for Systems and Control, Delft University of Technology, Delft, the Netherlands. The PhD research has been focused on distributed estimation, control, and optimization algorithms for robotic networks and has been supervised by Dr. T. Keviczky and Prof. dr. R. Babuška. During my PhD project I also graduated from the Dutch Institute for Systems and Control (DISC) in 2010, and I spent a three-month stay at the Automatic Control Lab of KTH, The Royal Institute of Technology, Stockholm, Sweden, collaborating with Prof. dr. M. Johansson and Dr. D. V. Dimarogonas.

“When one door closes another door opens;
but we often look so long and so regretfully upon the closed door,
that we do not see the ones which open for us.”

Alexander Graham Bell