Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Multi-Agent Distributed Optimization and Estimation over Lossy Networks

Ph.D. candidate
Nicoletta Bof

Advisor
Prof. Luca Schenato

Co-Advisor
Prof. Ruggero Carli

Director & Coordinator
Prof. Andrea Neviani

Ph.D. School in
Information Engineering

Department of
Information Engineering

University of Padova
2018

Abstract

Nowadays, optimization is a pervasive tool, employed in a lot different fields. Due to its flexibility, it can be used to solve many diverse problems, some of which do not seem to require an optimization framework. As so, the research on this topic is always active and copious. Another very interesting and current investigation field involves multi-agent systems, that is, systems composed by a lot of (possibly different) agents. The research on *cyber-physical systems*, believed to be one of the challenges of the 21st century, is very extensive, and comprises very complex systems like *smart cities* and *smart power-grids*, but also much more simple ones, like *wireless sensor networks* or *camera networks*. In a multi-agent context, the optimization framework is extensively used. As a consequence, optimization in multi-agent systems is an attractive topic to investigate.

The contents of this thesis focus on distributed optimization within a multi-agent scenario, i.e., optimization performed by a set of peers, among which there is no leader. Accordingly, when these agents have to perform a task, formulated as an optimization problem, they have to collaborate to solve it, all using the same kind of update rule.

Collaboration clearly implies the need of messages exchange among the agents, and the focus of the thesis is on the criticalities related to the communication step. In particular, *no reliability of this step is assumed*, meaning that the packets exchanged between two agents can sometime be lost. Also, the sought-for solution *does not have to employ an acknowledge protocol*, that is, when an agent has to send a packet, it just sends it and goes on with its computation, without waiting for a confirmation that the receiver has actually received the packet. Almost all works in the existing literature deal with packet losses employing an acknowledge (ACK) system; the effort in this thesis is to avoid the use of an ACK system, since it can slow down the communication step. However, this choice of averting the use of ACKs makes the development of optimization algorithms, and especially their convergence proof, more involved. Apart from robustness to packet losses, the algorithms developed in this dissertation are also asynchronous, that is, the agents do not need to be synchronized to perform the update and communication steps.

Three types of optimization problems are analyzed in the thesis. The first one is the *patrolling problem for camera networks*. The algorithm developed to solve this problem has a restricted applicability, since it is very task-dependent. The other two problems are more general, because both concern the *minimization of the sum of cost functions*, one for each agent in the system. In the first case, the form of the local cost functions is particular: these, in fact, are *locally coupled*, in the sense that the cost function of an agent depends on the variables of the agent itself and on those of its direct neighbors. The sought-for algorithm has to satisfy two properties (apart from asynchronicity and robustness to packet losses): the requirement of asking a single communication exchange per iteration (which also reduces the need of synchronicity) and the requirement that the communication among agents is only between direct neighbors. In the second case, the *local functions depend all on the same variables*. The analysis first focuses on the special case of local quadratic cost functions and their strong relationship with the consensus problem. Besides the development of a robust and asynchronous algorithm for the average consensus problem, a comparison among algorithms to solve the minimization of the sum of quadratic cost functions is carried out. Finally, the distributed minimization of the sum of more general local cost functions is tackled, leading to the development of a robust version of the Newton-Raphson consensus.

The theoretical tools employed in the thesis to prove convergence of the algorithms mainly rely on Lyapunov theory and the separation of scales theory.

Sommario

Oggigiorno l'ottimizzazione è uno strumento pervasivo, utilizzato in molti ambiti differenti. Grazie alla sua flessibilità, può essere utilizzato per risolvere numerosi problemi, alcuni dei quali non sembrano a prima vista poter essere formulati come problemi di ottimizzazione. Proprio grazie a questa versatilità, la ricerca sulle tecniche di ottimizzazione è sempre attiva e copiosa. Un altro tema di ricerca molto interessante e attuale riguarda i sistemi multi-agente, cioè sistemi composti da molti agenti (anche diversi fra di loro). La ricerca sui *sistemi ciberfisici*, ritenuta una delle sfide del ventunesimo secolo, è molto vasta, e comprende sistemi molto complessi come le *città intelligenti* e le *reti di potenza intelligenti*, ma anche quelli molto più semplici, come le *reti di sensori senza filo* o le *reti di telecamere*. In un contesto multi-agente, lo strumento dell'ottimizzazione è molto usato. Di conseguenza, l'ottimizzazione in sistemi multi-agente è un argomento attraente da investigare.

Questa tesi si concentra sull'ottimizzazione distribuita in uno scenario multi-agente, cioè in uno scenario in cui l'ottimizzazione è svolta da un insieme di entità con pari capacità, tra le quali non c'è un leader. Di conseguenza, quando questi agenti devono portare a termine una attività, formulata come un problema di ottimizzazione, devono collaborare per svolgerla usando tutti lo stesso tipo di azioni.

La collaborazione chiaramente richiede lo scambio di messaggi tra gli agenti, e in questa tesi l'attenzione è focalizzata sulle criticità relative alla fase di comunicazione. In particolare,*non si assume che la comunicazione sia affidabile*, e di conseguenza i pacchetti (cioè i messaggi) scambiati tra due agenti possono essere talvolta persi. Inoltre, la soluzione ricercata *non deve richiedere un protocollo di conferma*, cioè, quando un agente deve inviare un pacchetto semplicemente lo invia e continua con le sue computazioni, senza aspettare la conferma che l'agente a cui ha inviato il pacchetto lo abbia effettivamente ricevuto. Quasi tutti i lavori esistenti in letteratura gestiscono le perdite di pacchetto utilizzando un sistema di conferma; lo sforzo in questa tesi è proprio quello di evitare l'uso di messaggi di conferma, dal momento che questi possono rallentare la fase di comunicazione. Questa scelta rende però lo sviluppo di algoritmi di ottimizzazione, e

specialmente la dimostrazione della loro convergenza, più complicati. Oltre alla robustezza alla perdita di pacchetti, gli algoritmi sviluppati in questa tesi sono anche asincroni, cioè gli agenti non devono necessariamente essere sincronizzati per svolgere le fasi di aggiornamento e comunicazione.

In questa tesi vengono analizzati tre tipi di problemi di ottimizzazione. Il primo è il problema della perlustrazione effettuata da reti di telecamere. L'algoritmo sviluppato per questo problema ha una applicabilità limitata, essendo molto legato al problema stesso. I rimanenti due problemi sono più generali, e riguardano la minimizzazione della somma di funzioni costo, una per ogni agente nel sistema. Nel primo problema, la forma delle funzioni costo locali è particolare. Le funzioni costo sono infatti localmente accoppiate, nel senso che la funzione costo di un agente dipende dalla variabile dell'agente stesso e da quelle dei suoi vicini diretti. L'algoritmo ricercato deve soddisfare due richieste (oltre alla asincronia e alla robustezza alla perdita di pacchetti): deve richiedere un solo scambio di messaggi per iterazione (per ridurre la necessità di sincronizzazione) e deve richiedere lo scambio di informazioni solo tra vicini diretti. Nel secondo problema, le funzioni locali dipendono tutte dalle stesse variabili. L'analisi in questo caso prima si focalizza sul caso speciale di minimizzazione di funzioni costo quadratiche e la loro forte relazione con il problema di consenso. Oltre allo sviluppo di un algoritmo robusto e asincrono per il problema del consenso alla media, viene anche svolta una comparazione tra diversi algoritmi che risolvono la minimizzazione della somma di funzioni costo quadratiche. Infine viene affrontata la minimizzazione distribuita della somma di funzioni costo locali più generali, portando allo sviluppo di una versione robusta del Newton-Rapshon consensus.

Gli strumenti teorici impiegati in questa tesi per provare la convergenza degli algoritmi sono soprattutto la teoria di Lyapunov e la teoria della separazione delle scale temporali.

Contents

1 Introduction

1.1 Multi-agent systems

Nowadays, the presence of the so-called *smart systems* is becoming more and more pervading. The concept of smart system is no more only known by people working in the research world, but it is also a topic of interest for magazines and newspapers.

An example are *smart cities* Albino, Berardi, and Dangelico (2015). They can be thought as cities where the different systems (communication, transportation, electricity, infrastructures, and so on) share the information collected by their own sensors, and join efforts to improve the sustainability of the cities themselves but also the quality of life of its citizens. Smart cities are the protagonists of many articles: most of them praise their promise to guarantee a safer, healthier and more efficient and sustainable environment, which clearly makes them appealing and interesting for everyone Kotkin (2009); Balch (2013); Singh (2014); Wheeland (2016); Totty (2017). However, some pieces appearing in the magazines also highlight some drawbacks which may affect these intelligent cities Perlroth (2015). Figure 1.1 shows how many different systems have to work together to realize a smart city, while Figure 1.2 shows some of the smart city projects in development.

Clearly, *smart buildings* will be a fundamental actor in the development of smart cities. All the different available services in a building will work together to offer a better

Figure 1.1: Different systems which are involved in the creation of a smart city.
Credits: https://smartcitiesworld.net/news/news/smart-cities-services-worth-225bn-by-2026-1618 smart city

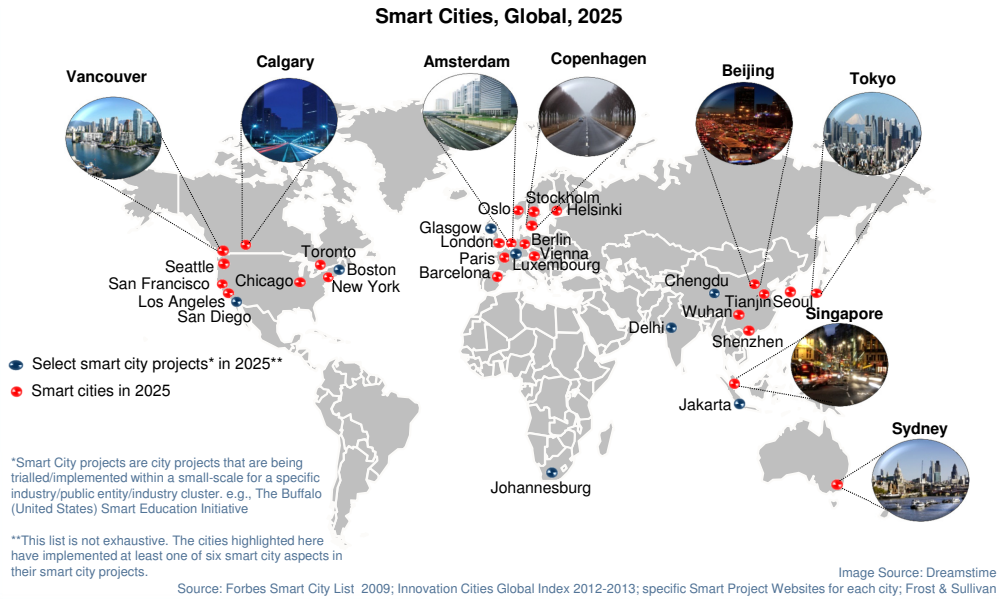


Figure 1.2: Some of the smart city projects around the world.
Credits: http://www.egr.msu.edu/ aesc310-web/resources/SmartCities/Smart%20City%20Market%20Report%202.pdf

living experience (see Figure 1.3). For example *smart house appliances* will coordinate their daily use of the electric power in order to reduce the overall cost (thanks to the time-varying price of the electricity) Sou, Weimer, Sandberg, and Johansson (2011), offering at the same time a better experience to the house's inhabitants Cook, Youngblood, and Das (2006). The security of such systems is of paramount importance, due to their vulnerability to hacker attacks Twentyman (2017).

Figure 1.3: Different actors involved to make a building smart.
Credits: https://cleantechnica.com/2012/11/30/ibm-helping-uk-ministry-of-defence-go-smarter-and-greener/

Also *smart traffic systems* have recently acquired popularity Sanburn (2015); Quain (2017); Cardwell (2017), due to the possibility they offer to reduce congestions. They consist of systems where, among the others, traffic sensors, traffic lights and also cars can share information, allowing for a better management of the traffic. Reduction of traffic is clearly an appealing notion, since it implies reduction of both pollution and trip duration. An example of the complexity of a smart traffic system can be found in Figure 1.4.

Figure 1.4: A Smart Traffic System.
Credits: http://www.brindleytech.com/smart-transportation/

Connected with smart transportation, also smart *autonomous cars* are envisioned to be part of the future of the transportation system Dresner and Stone (2008). Safety and reliability of the trips will be of paramount importance for these systems. Figure 1.5 shows the autonomous cars developed by Google. Many newspapers' articles can be found on this topic Horton (2017); Colias and Higgins (2017); Curtis (2017).

Figure 1.5: Google's autonomous cars.
Credits: https://plus.google.com/+SelfDrivingCar

Another concept, *smart power grids* Ipakchi and Albuyeh (2009); Rogers, Ramchurn, and Jennings (2012), is quite popular for magazines and newspapers Walsh (2009); Cardwell (2015); O'Boyle (2017). An intelligent version of the power grid can allow to better manage blackouts, power losses along the lines and, especially, to better exploit the power produced by renewable sources like solar panels and wind power plants. Figure 1.6 presents some of the players in such a system.

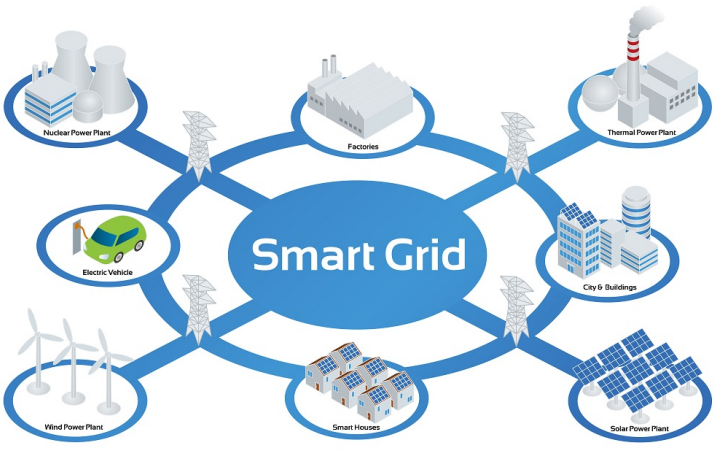


Figure 1.6: Some of the possible participants in a smart power grid.
Credits: https://www.enisa.europa.eu/media/press-releases/smart-grid-security-certification-in-europe-challenges-and-recommendations

Finally, also the term *cyber-physical system* (CPS) Kim and Kumar (2012) conquered newspapers Marr (2016b); Poole (2017). The definitions of this concept are usually very broad, like the following one *"The term cyber-physical systems refers to the tight conjoining of and coordination between computational and physical resources. We envision that the cyber-physical systems of tomorrow will far exceed those of today in terms of adaptability, autonomy, efficiency, functionality, reliability, safety, and usability. Research advances in cyber-physical systems promise to transform our world with systems that respond more quickly (e.g., autonomous collision avoidance), are more precise (e.g., robotic surgery and nano-tolerance manufacturing), work in dangerous or inaccessible environments (e.g., autonomous systems for search and rescue, firefighting, and exploration), provide large-scale, distributed coordination (e.g., automated traffic control), are highly efficient (e.g., zero-net energy buildings), augment human capabilities, and enhance societal wellbeing (e.g., assistive technologies and ubiquitous healthcare monitoring and delivery)."* NSF (2008), and as so, a cyber-physical system subsumes all the previous systems presented, and includes also, for example, Factory 4.0 projects Bryant (2014); Marr (2016a). Clearly, due to all the envisioned benefits of such a technology, CPSs are considered one of the challenges of the 21st century and lot of effort is devoted to research on this topic Kim and Kumar (2012); Esterle and Grosu (2016).

The figures presented above well show how many different entities are involved in a smart system or in a CPS. In all these, it is possible to identify different agents that collect and share information, perform some computation and then decide an action to take. Consequently, they are all example of very complex multi-agent systems.

A *multi-agent system* is composed by a number of independent smart agents (or nodes) which can interact among themselves and can possibly collaborate to perform some tasks. These kind of systems have become more and more permeating in everyday life (even without considering the previous recalled complex smart systems), due to technological advancement. As a matter of fact, it is nowadays possible to produce micro-processors with significant computational capability at a small price. The employment of such processors allows to cheaply make much smarter sensors and robots, together with a number of different every day objects (for example house appliances).

Examples of multi-agent systems range from systems involving very small agents to those involving very big entities, from systems involving agents with a lot of different capabilities to those composed of agents with just sensing capabilities (apart from the computational power), and also the aim of these systems can be very different. The examples presented at the beginning of the chapter involve multi-agent systems that are currently being designed, and their full development and use will happen in the future. In

the following, some other examples of multi-agent systems are given. These will involve systems that are already deployed and utilized nowadays, and some are also of interest for the applications studied in this manuscript.

Wireless sensor networks (WSN) Estrin, Girod, Pottie, and Srivastava (2001); Puccinelli and Haenggi (2005), see Figure 1.7, are a good example of multi-agent systems. They are made up by a very large number of (possibly small) sensors (the agents) which are deployed in a given area in order to collect some measurements of interest. The fusion of all the sensors' data is the main task the agents have to perform.

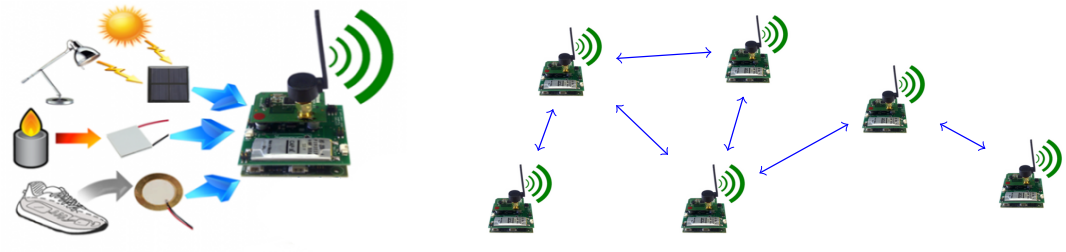


Figure 1.7: On the left, a wireless sensor with different sensing capability (for light, temperature and pressure). On the right, a wireless network composed of such sensors. A blue arrow between two agents means that these agents can communicate
Credits: https://www.elprocus.com/architecture-of-wireless-sensor-network-and-applications/

Smart camera networks Aghajan and Cavallaro (2009) are multi-agent systems made up of smart cameras which monitor the environment through vision sensing. The tasks of these networks range from patrolling to intruders tracking and anomaly detection. Figure 1.8 shows an example of camera network.

Other interesting multi-agent systems are those made of autonomous robots. These systems can perform tasks of environment monitoring but can also act on the environment (depending on the robots' capability) Portugal and Rocha (2011); Iocchi, Nardi, and Salerno (2000).

As in many other fields, also for multi-agent systems the optimization framework is a very effective tool. As a matter of fact, the majority of the tasks to be solved can be formulated as an optimization problem, or at least an optimization framework can be adopted, in order to improve the performance of the overall system. For example, the patrolling problem for a camera network can be formulated as the optimal scheduling of the cameras' movements in order to minimize the time between two consecutive visits of each point of the perimeter. The problem of estimating a quantity given some measurements of the same, can be seen as the minimization of the norm of the difference between the measurements and the quantity.

On the other hand, assuming that each agent in the system has some computational

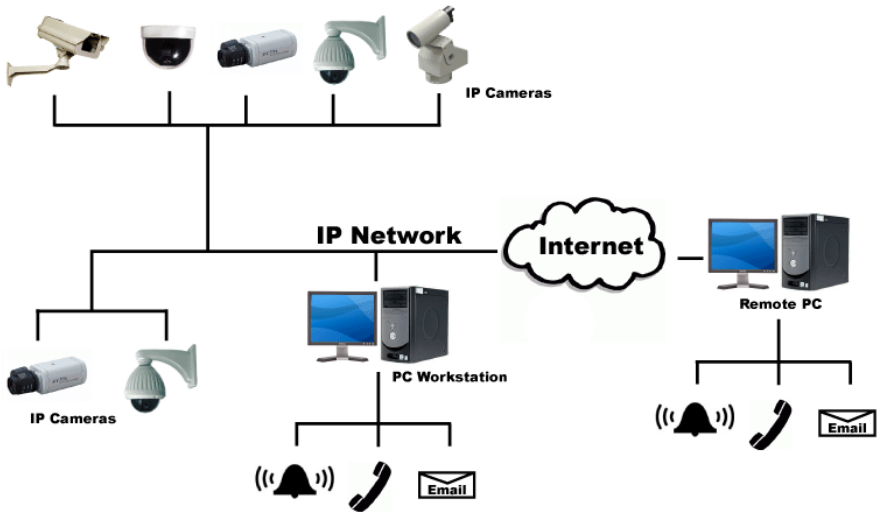


Figure 1.8: Smart camera network for security purpose.
Credits: http://www.dynapost.com/index.php?page_no=33

power, the optimization problem defined to accomplish the task can be solved in a distributed way by the agents themselves. As a consequence, multi-agent systems are very good test-beds for distributed optimization algorithms, in particular for those with a peer-to-peer architecture. Assume, in fact, that there is no hierarchy among the agents. Then, the algorithms which can be run on these systems have to assume that all the agents can only perform the same kind of steps in order to solve the problem. The development of such distributed algorithms has some peculiarities with respect to that of centralized ones. In particular, one has to keep in mind that (depending on the system's agents) the computational and memory capability of each node might be limited, and, more importantly, that the communication between agents is fundamental to perform the optimization. The work presented in this thesis focuses on distributed algorithms, both for quite specific problems and for more general ones, with an explicit attention to (some) real-world communication issues.

1.2 Distributed optimization

Optimization is a very important branch of knowledge, applied to a vast variety of different fields. Examples of its application can be found in engineering in the design of electronic systems (*device sizing*) or in the development of controllers (*optimal control*), in economics for the maximization of profits (*portfolio optimization*) Boyd and Vandenberghe (2004), in a firm's management to plan production and storage Chazal, Jouini, and Tahraoui (2008). Due to its versatility, much study has been conducted on this topic from many

diverse points of view Bertsimas and Tsitsiklis (1997); Boyd and Vandenberghe (2004); Bertsekas (2014); Horst, Pardalos, and Van Thoai (2000) .

Given an optimization problem related to a multi-agent system, this might be solvable in a centralized manner, assuming that all the data involved in this problem could be stored in a single central computational unit. In fact, under this assumption, all the computation can be done in the central unit and then the result can be sent back to the agents. Figure 1.10 gives a pictorial representation of the problem resolution procedure using a centralized algorithm in a multi-agent set-up. Many algorithms have been developed to solve optimization problems using a centralized approach, and this is still an important branch of research (see the references above).

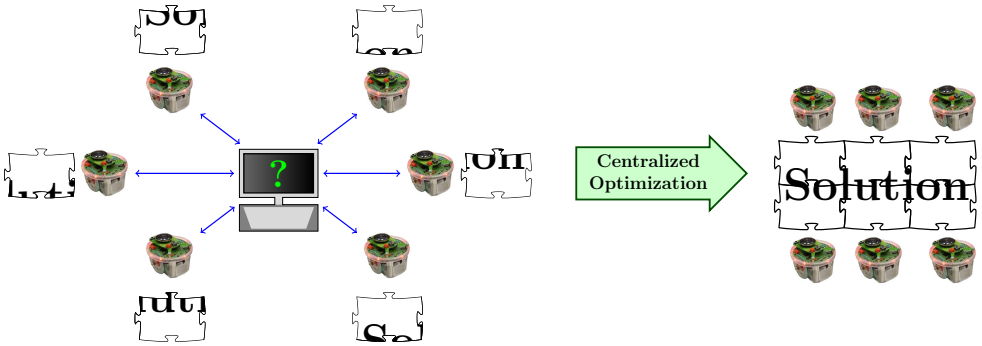


Figure 1.9: Starting from the left, each agent has its own part of the problem, and the possibility to share information with a central unit. The overall problem is solved using a centralized algorithm run by the central unit, and then the solution, communicated to the agents by the central unit, is employed by the agents to "physically" solve the problem.
Credits: the puzzle design has been made thanks to http://www.texample.net/tikz/examples/puzzle/
the mini-robot has been obtained from http://mobots.epfl.ch/e-puck.html

However, in a multi-agent framework with smart agents, such an approach would be wasteful. In fact, the computational power of each node would not be exploited. Moreover, in a multi-agent system, the problem itself might be in a way distributed and with the increasing privacy-related issues, for example in smart power grids Quinn (2009), the collection of all the data in a unique center might be problematic. All these facts suggest that, in a multi-agent system, a distributed approach might be more suitable (if not indispensable). Figure 1.10, shows how a problem is solved using a distributed approach, as opposed to the centralized approach showed in Figure 1.9.

The importance and spread of distributed algorithms is not exclusively due to their application in multi-agent systems. As a matter of fact, a leading field in this sense is also the Big Data research area, where the huge quantity of data to be analyzed makes a centralized approach infeasible Dobre and Xhafa (2014); Boyd, Parikh, Chu, Peleato, and Eckstein (2011). In this case, the solution of such problems in a centralized way

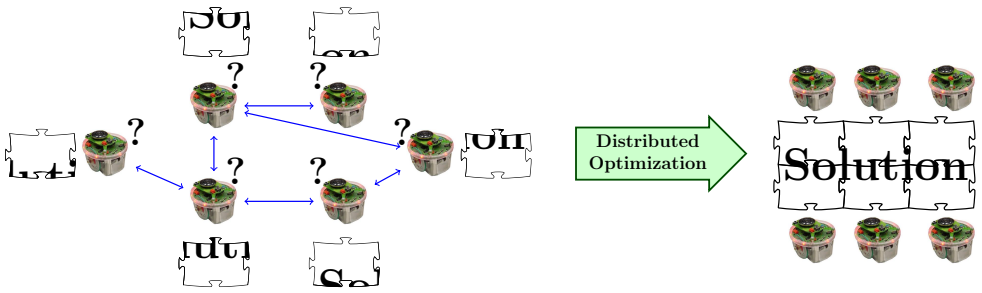


Figure 1.10: Starting from the left, each agent has again its own part of the problem, but only the possibility to share information with its own neighbors. The problem is then solved by each agent using a distributed algorithm. After each agent evaluates the optimal solution, it employs this solution to "physically" solve the problem.

would require very high computational capability together with huge memory quantity, making the solution very expensive and difficult to manage and implement. Distributing the problem among different computational unit is usually the solution adopted to solve such big problems.

Due to these different causes, a lot of research has been dedicated to the development of distributed algorithms, starting approximately forty years ago Tsitsiklis (1984); Bertsekas and Tsitsiklis (1989).

As already pointed out, one peculiarity of distributed approaches is that the agents performing the optimization have to exchange information. This is also one significant difference between a centralized and a distributed approach. As a matter of fact, in a centralized approach one iteration of the algorithm just requires the update of the variables involved, while in a distributed one, an iteration is also made of a communication step, see Figure 1.11.

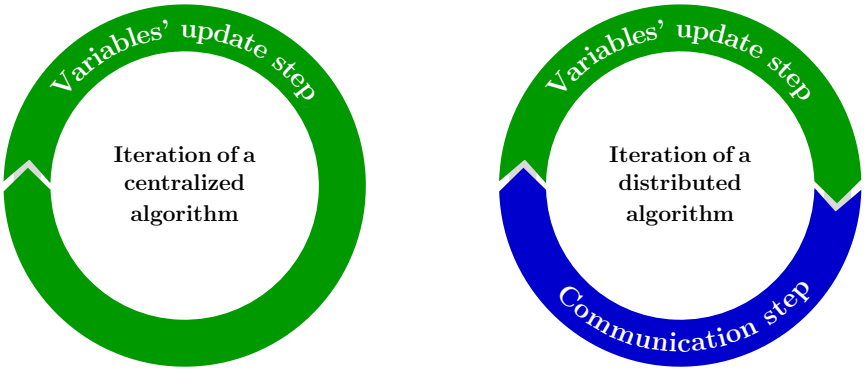


Figure 1.11: Differences between one iteration of a centralized algorithm and one of a distributed algorithm.

In order to perform an optimization in a distributed way, the communication between agents is of paramount importance (here it is assumed that each agent does not have all the information to evaluate an optimal solution by itself). In fact, communication allows the different agents to share some kind of information in order to reach an optimal solution of the problem. Without communication, each agent would just try to solve the problem at the best of its own knowledge, and so it is improbable that an optimal solution for the whole problem is reached.

In the development of a distributed algorithm, the communication's features are very important. One of these features is the connectivity among agents, that is with which other agents a given agent can communicate. For a given problem for example, the speed of the algorithm in retrieving an optimal solution when each agent communicates with all the others can be very different with respect to a case where each agent can communicate only with the two nearest neighboring agents. The two different situations are described in Figure 1.12. In some cases, it is possible to show that the choice of the algorithm should be guided also by this communication feature Bof, Carli, and Schenato (2016b).

Figure 1.12: On the left, a multi-agent system where all the agents communicate with all the others. On the right, a system where an agent communicates only with the two nearest neighbors. The convergence speed of the same distributed algorithm in these two different situations can be dramatically different.

However, more important are the features concerning the *reliability* of a real-world communication network. A lot of existing distributed algorithms assume that the communication is reliable, that is no packets are lost during communication. This might not be accurate at all, especially if the communication network is wireless (which is common practice in a lot of multi-agent systems). The consequences of packet losses can be detrimental for the solution of an optimization problem. Figure 1.13 shows the differences between a reliable and an unreliable communication network.

Some works in the literature deal with packet losses assuming an *acknowledge system*, or the agent's ability to test in advance the availability of the communication channel (even though this assumption is sometime not clearly stated) Kar and Moura (2009, 2010); Patterson, Bamieh, and El Abbadi (2007). If an acknowledge system is used, the agent which sends the message considers the exchange completed only upon reception of

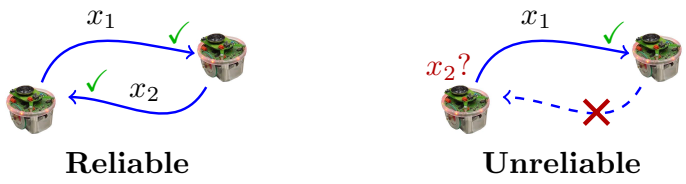


Figure 1.13: On the left a communication exchange between two neighbors, with both exchanges that are successful. On the right, the packet going from the agent on the right is not received by the agent on the left, which, as a consequence, does not know how to act.

an acknowledge message (or ACK) from the receiver, and in the meanwhile it periodically re-sends the message. Figure 1.14 illustrates the operation of an acknowledge system. As will be later described in details, these kind of solutions have some disadvantages and it might be preferable to develop distributed algorithms inherently robust to packet losses, which are denoted as *robust* in this dissertation. In particular, from Figure 1.14 it is clear that one of the disadvantages of using an acknowledge system is that it can slow down the attainment of the optimization problem's solution, since the communication step can last for a consistent duration of time.

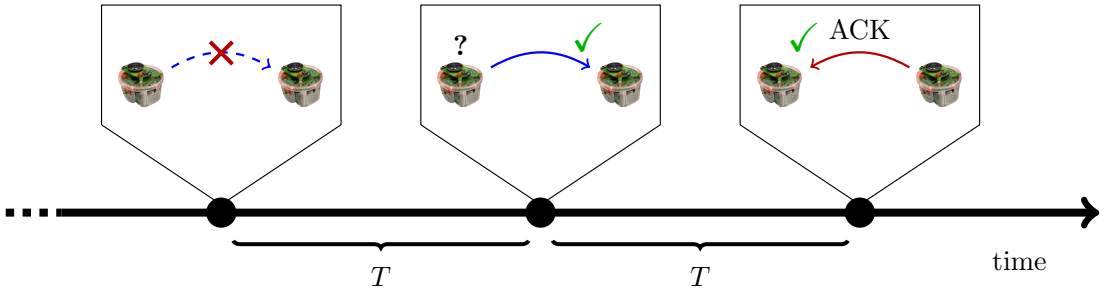


Figure 1.14: Operation of an acknowledge system in a communication exchange. The first message sent is lost, so after an interval of duration T, the sender, not having received the ACK, sends the message again. This second time the exchange is successful and after another interval of duration T the sender receives the ACK and considers the communication exchange completed.

Another aspect related to the communication is whether the algorithm needs a synchronous message exchange. Namely, some algorithms, denoted as synchronous, need all the agents to communicate at the same time. As a consequence, they need either a method to detect if they have all reached the communication step before performing the communication or they need a global clock in order to be synchronized. On the other hand, an algorithm is asynchronous if it does not require this synchronization among agents. For example, at each time step there might be just a single node that performs some computation and then sends the related information to its neighbors.

Figure 1.15 presents the two different kinds of communication. The need for synchronous communication can slow down the algorithm, and can also be difficult to satisfy when the number of agents in the network increases.

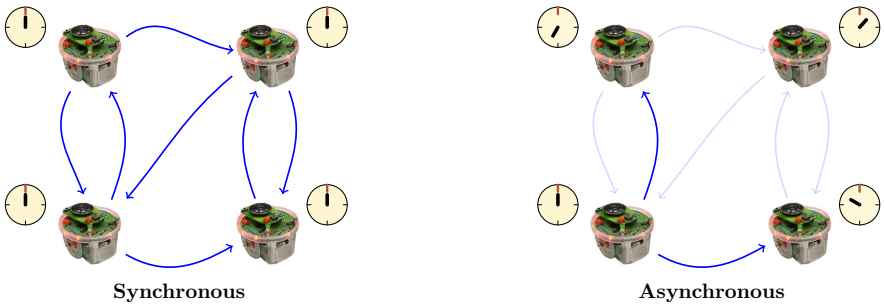


Figure 1.15: On the left, a synchronous communication exchange with all the agents' clocks synchronized on the same time. On the right, asynchronous communication exchange with only one agent performing the communication step. In this case all the agents have a possibly different time reference.

In this work the aforementioned communication's features will be in different detail explored. However, there are also other aspects of real-world communication which are important, among which there are noisy communication channels and quantization errors. Kar and Moura (2009, 2010) for example deal with such problems in the specific case of a group of agents that have to reach consensus on the mean of some quantity, that is the average consensus problem.

1.3 Challenges & contributions

Most of the work presented in this thesis focuses on the development of algorithms to solve different kinds of problems and that are able to cope with an unreliable and asynchronous communication scenario. The communication network that is used does not employ an acknowledge system and this allows to avoid the related disadvantages. However, accepting the possibility of packet losses renders the solution of a problem in a distributed set-up more challenging. A very famous example of this fact is the Witsenhausen counterexample Witsenhausen (1968), which will be later described.

Most of the convergence proofs for the algorithms rely on stability theory for (time-varying) dynamical systems. As a matter of fact, an algorithm iteration can be seen as a (possibly time-varying) dynamical system, and for such systems many techniques can be used to prove stability: Lyapunov functions will be highly exploited (in Chapters 3, 4 ,6), together with separation of time scale theory (Chapters 4, 6), but also stability theory for linear systems (Chapter 5).

Four different problems are tackled in the thesis, and different are the solutions found in order to cope with the imperfect communication scenario. The ideas which allows to deal with asynchronous and lossy communication are diverse, and it is interesting to highlight the differences between these ideas. In order to do so, assume that, for each problem analyzed, there exists a distributed algorithm which is able to solve the problem if the communication scenario is reliable and synchronous. As a consequence, it is possible to assume that every node in the system has an estimation of the problem's solution and a mechanism to apply in order to make this estimate better, fact which is represented in Figure 1.16, for a very basic system made of two robots. Starting from such an algorithm, one can try to look for ways to obtain an asynchronous and robust algorithm.

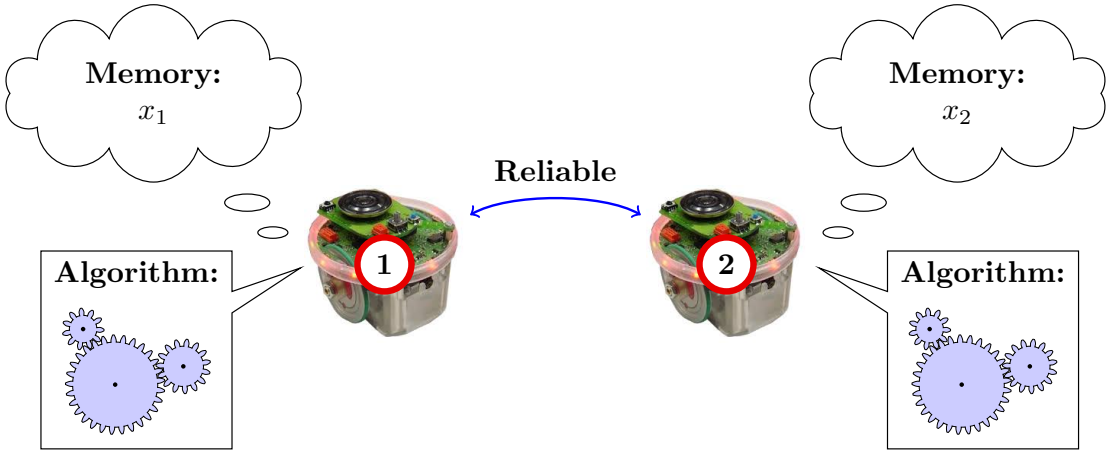


Figure 1.16: Given the problem in case of reliable communication, it is assumed that the agents know how to act. x_1 and x_2 represents the tentative solution of agent 1 and 2 respectively, and the gears represents the method (the algorithm) using which the agents can improve their own tentative solution until it becomes good enough.

Always using Figure 1.16 as a comparison, the different problems tackled in the thesis are here briefly introduced, with a special attention on why it can be seen as an optimization problem (if not clearly so) and on the peculiarity of the solution which allows to cope with imperfect communication.

Patrolling problem

The first application, analyzed in Chapter 3, concerns the patrolling problem. This problem arises in wireless smart camera networks (see Figure 1.8), as well as in robot networks. Patrolling is the act of constantly monitoring an area in order to detect

anomalies or intruders (see Figure 1.17). It can be formulated as an optimization problem in order to improve the performance of the monitoring. As a matter of fact, very easy solutions can be promptly found if one is just interested in having the area monitored every now and then, but the quality of the monitoring can be highly enhanced if an optimization framework is employed.

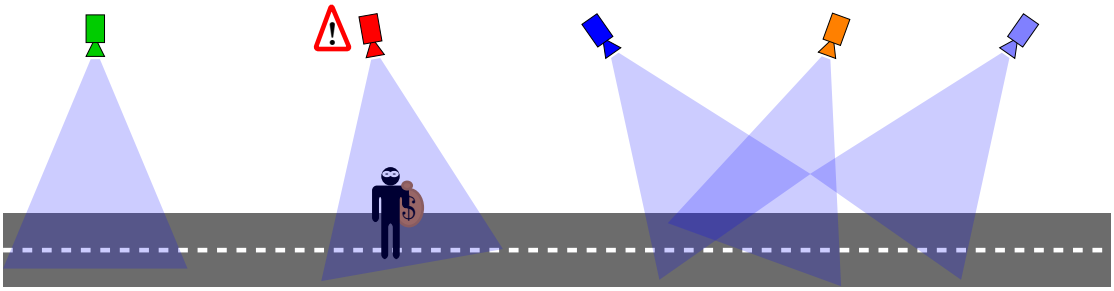


Figure 1.17: A video surveillance system which monitors a street. The cameras patrol the street and an alarm turns on if a suspicious event is detected, like in the case of the red camera which pinpoints a thief.

After developing an approach in case of a reliable communication scenario, this algorithm is modified in order to cope with imperfect communication. In this first case, the winning idea is a modification of the starting algorithm, see Figure 1.18. Even though the modification applied to obtain the robust algorithm is small, the proof of convergence for this new algorithm is more involved than the proof for the starting one. The algorithm developed is particularly tailored for the problem itself, and, as a consequence, is not easily applicable to other problems.

Locally coupled cost

Chapter 4 is dedicated to a more general problem, the minimization of the sum of local cost functions which are locally coupled. Such problems can arise in different kinds of multi-agent systems, where each agent has a private cost function, that depends on variables that either strictly belong to the agent itself or to one of its neighbors. The function is locally coupled because each agent needs only information from its neighbors to evaluate it. Figure 1.19 visualizes this concept of locally coupled function.

In this second case, the idea behind an algorithm which works in an asynchronous and lossy communication scenario is developed starting from an idea similar to Newton's method, which is a centralized optimization algorithm, later described in Chapter 2. First a synchronous distributed version is obtained, which works in a reliable communication set-up, and later a robust version of the algorithm is developed. In this case, the trick

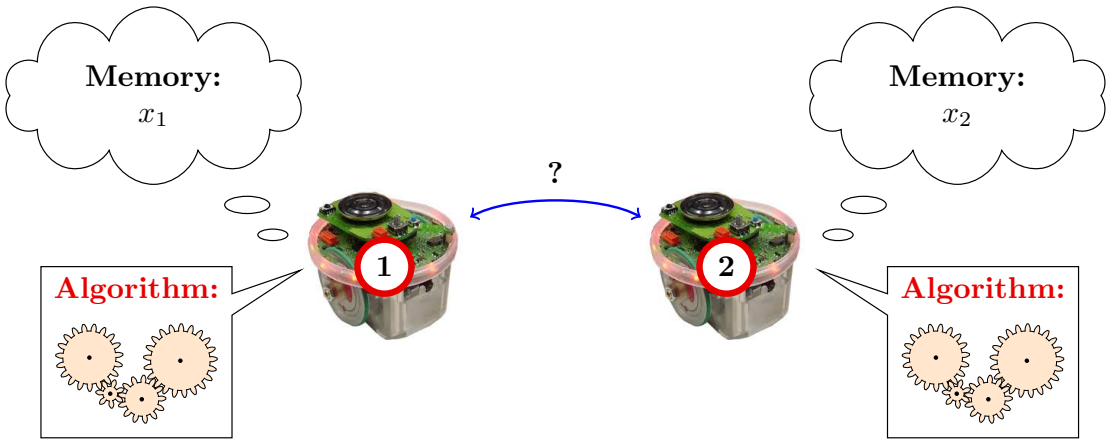


Figure 1.18: In order to deal with imperfect communication, one method used in the thesis is the modification of the starting algorithm (which is only able to deal with perfect communication). See Figure 1.16 for a comparison.

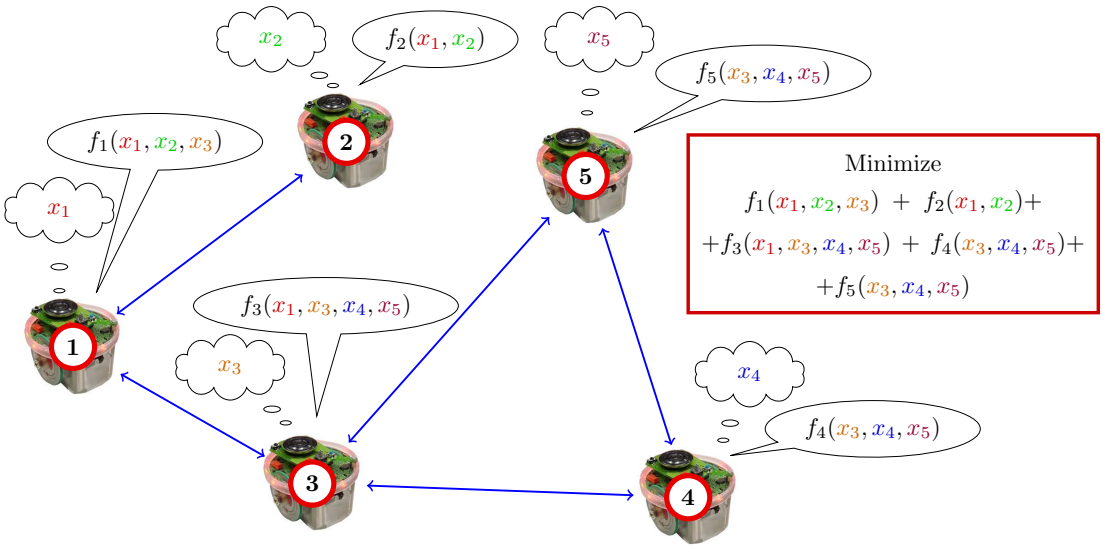


Figure 1.19: Each agent in the system has its own function f_i, which depends on its variable x_i and on those of its neighbors (indicated by the blue arrows). The aim of each agent is to minimize the sum of these functions.

that allows the operation in a non-ideal communication scenario is the use of memory (that is each agent remembers the last message received from its neighbors). This trick is useful also because it allows a real distributed approach, as will be shown in Chapter 4. Figure 1.20 visualizes the difference between the (respective) original algorithm.

The algorithm developed in this chapter is employed for a smart power grid application,

in particular for voltage state estimation from current and voltage measurements. However, it is possible to apply this algorithm also for localization in sensor networks or in multi-robots formation.

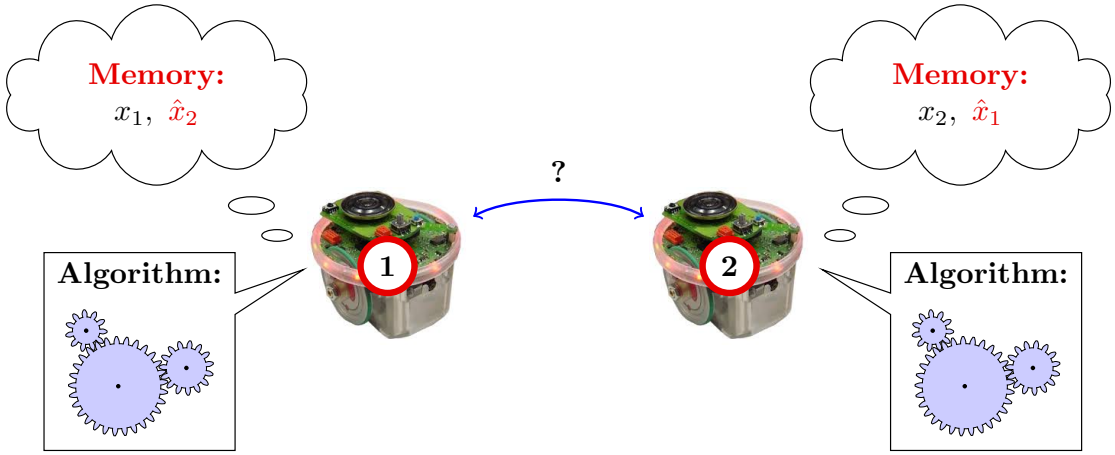


Figure 1.20: In order to deal with imperfect communication, the second method used in the thesis is allowing each agent to keep in memory the last message received from its neighbors. In this case, agent 1, apart from keeping in memory its own estimate x_1, also keeps in memory $\hat{x}_2$, which is the last estimate x_2, made by agent 2, that agent 1 has received. Every time agent 1 receives a new message from agent 2, $\hat{x}_2$ is updated. Agent 2 acts similarly. See Figure 1.16 for a comparison with the starting algorithm.

Consensus problem and quadratic cost minimization

Chapter 5 is devoted to the average consensus problem. According to this problem, each agent is the owner of a private quantity, and it want to independently evaluate the mean of all the individual quantities (Figure 1.21 is an example where consensus can be useful). This problem is strictly related to the distributed minimization of the sum of quadratic cost functions. If one is able to distributely solve one of these two problems, he can also solve the other. Due to this minimization interpretation, consensus is a problem of interest for this thesis.

A first part of the discussion on consensus problems is devoted to the study of the convergence rate of well known distributed algorithms (non-robust and synchronous) that either solve the consensus problem or the minimization of the sum of quadratic functions. In particular, the focus of the analysis is on the influence of the communication graph connectivity (see Figure 1.12) and of the curvature of the quadratic functions (according to the interpretation of consensus as the minimization of quadratic cost functions).

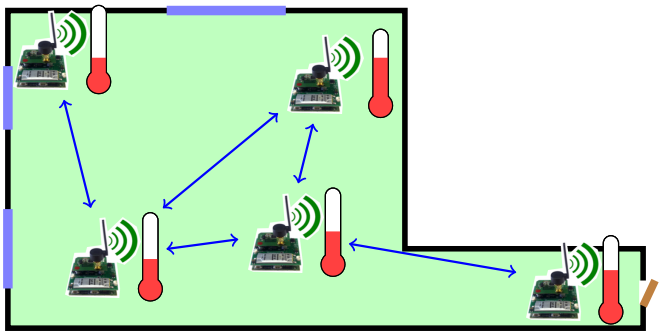


Figure 1.21: A group of small wireless sensors are located in a room in order to measure its temperature. Due to measurement errors, the detected temperature is different among the nodes, and a more accurate estimate of this quantity is given evaluating the mean of the measurements.

A second part of the discussion is dedicated to the development of an asynchronous and robust version of the average consensus algorithm. This algorithm is obtained as a combination of two algorithms already existing in the literature, one which is asynchronous but not robust and one which is robust but synchronous. In the robust algorithm developed, the idea which allows robustness is both an algorithm modification and a state augmentation. Namely, starting from the asynchronous consensus algorithm, the introduction of new variables (which are not merely variables to keep in memory the last packet received from the neighbors) and the consequent modification of the algorithm give a robust algorithm. Considering in this case the asynchronous algorithm as the starting one (whose representation is given in Figure 1.16), the new algorihtm can be represented as in Figure 1.22.

Additively separable cost function

The last application examined in this dissertation involves the minimization of additively separable cost functions. In particular, given a multi-agent system where each agent is endowed with a private cost function, the aim for each agent is to evaluate the minimizer of the sum of the cost functions. Differently from the case of locally coupled functions, here all the local cost functions depend on the same optimization variables. As a consequence, all the agents want to agree on the choice of the minimizer. Figure 1.23 is a pictorial representation of the problem to be solved. This problem surmises the consensus one (due to its interpretation as a minimization of sum of quadratic cost functions).

The aim of Chapter 6 is to solve the minimization of additively separable cost functions developing an algorithm which is asynchronous and robust to packet losses. The starting point for the development is an already existing algorithm, the Newton-

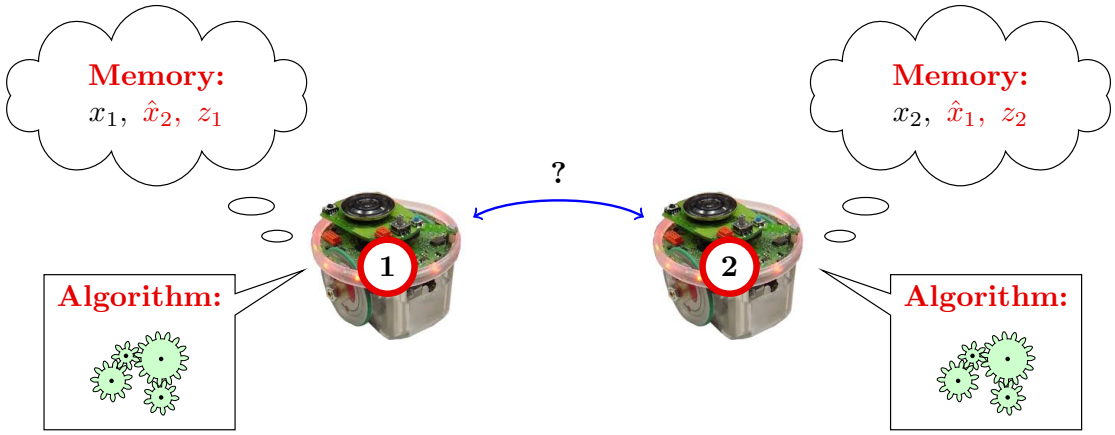


Figure 1.22: The third possible method used in order to deal with imperfect communication, entails a modification of the algorithm, the use of previously received messages from the neighbors and the introduction of new variables z_1 and z_2 respectively for agents 1 and 2. Figure 1.16 is again the term of comparison.

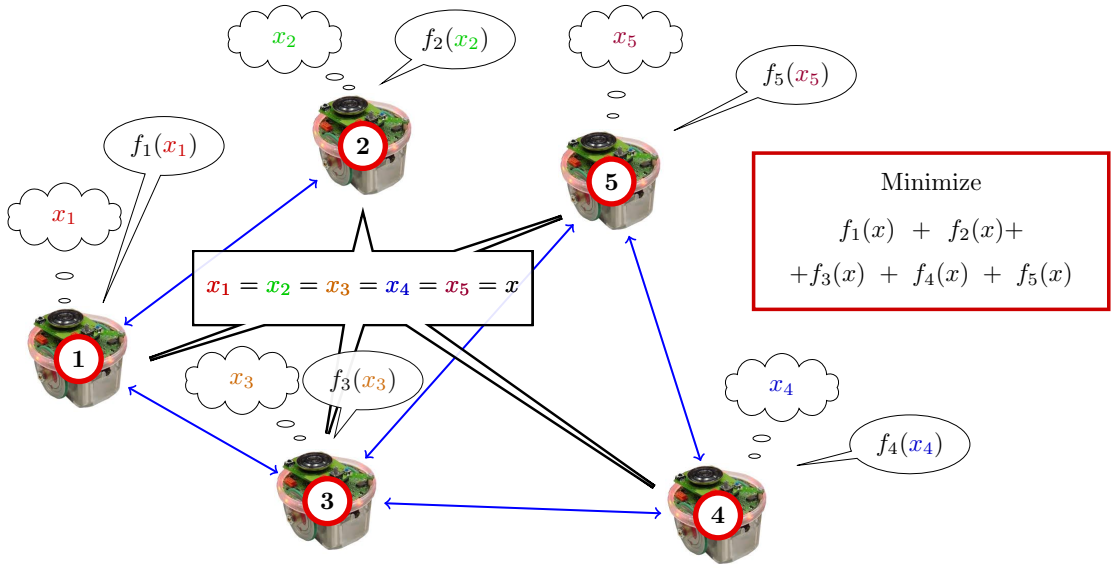


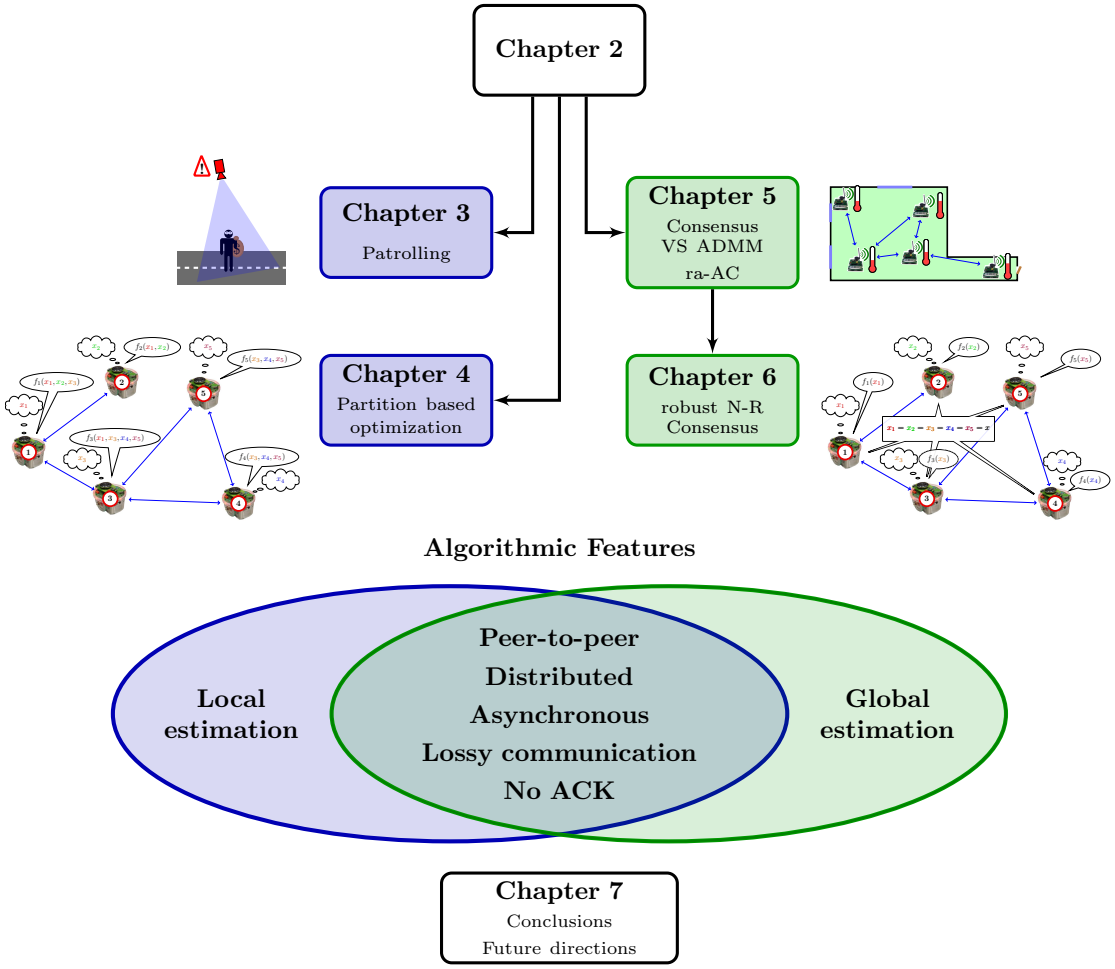
Figure 1.23: Like in Figure 1.19, each agent has its own cost function f_i, but differently from the same figure, here all the agents wants to agree on the choice of the variable. Again, each agent wants to minimize the sum of the private cost functions.

Raphson Consensus. In this algorithm, the only step requiring information's exchange among neighbors is a consensus step. The robust and asynchronous version of the Newton-Raphson Consensus is obtained employing the robust and asynchronous version of consensus developed in the previous chapter of the thesis. The proof of convergence is

more challenging than that of the starting algorithm. In particular, some quantities do not converge, even when the minimizer reaches the optimal point.

1.4 Manuscript outline

The outline of this manuscript is graphically shown below. Chapter 2 contains an introduction useful for all the remaining chapters. Chapters 3 and 4 can be read on their own, while Chapters 5 and 6 need to be read in order. The illustration also shows the features of the algorithm developed in each chapter (some of the keywords will be thoroughly discussed in Chapter 2). Finally, Chapter 7 draws the conclusions and describes some possible future directions.



Some mathematical preliminaries, symbols and notation useful in order to read the thesis can be found in Appendix A.

2

Distributed optimization

After pointing out in the introduction the wide applicability of optimization in man fields, this chapter formally introduces the optimization framework. In particular, it will focus on unconstrained convex minimization, since the problems analyzed in the remaining chapters mainly belong to this class. Most of the notions of this first part of the chapter are extracted from Boyd and Vandenberghe (2004). Then, the meaning of distributed approach will be explored, together with some state of the art in distributed optimization. This chapter also introduces the challenges related to communication between agents in a distributed approach. It finally closes with a brief description of the different problems studied in the thesis, to highlight the peculiarities of each application and of the methods proposed.

2.1 Optimization problems

An optimization problem is made of three ingredients, a function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, a set $\mathcal{X} \subseteq \mathbb{R}^n$ where the *optimization variable* $\boldsymbol{x}$ which minimizes $f(\boldsymbol{x})$ has to be searched, and a task, whether the function has to be minimized or maximized. Since the maximum of a function f corresponds to the minimum of $-f$, here the task is always the minimization of the function. In view of this choice, function f will be called cost function.

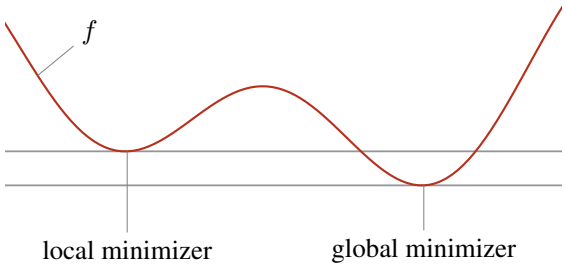


Figure 2.1: A function with a local and a global minimizer.

Mathematically, an optimization problem can be written as

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{2.1}$$

and its solution requires to determine the minimum value that f can achieve over the set $\mathcal{X}$. In general, more than finding the value of the minimum, one is interested in the elements (there might be more than one) in $\mathcal{X}$ that minimize the function, that is to find (at least one) $\boldsymbol{x}^* \in \mathcal{X}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \forall\ \boldsymbol{x} \in \mathcal{X}$. Such an $\boldsymbol{x}^*$ is called minimizer. Recalling some of the examples given in the introduction, in optimal control one is interested in determining a controller which minimizes a given cost function. This cost function might for example quantify the control effort and guarantee some performance. In portfolio optimization, on the other hand, one is interested to know what are the investments that will produce the best profit. Note that the minimizer previously defined is a *global* minimizer, but for a general function f it is possible to also have local minimizers. For clarity, a formal definition of both is now given (Figure 2.1 graphically shows the difference).

Definition 2.1.1. A vector $\boldsymbol{x}^* \in \mathcal{X}$ is a *local minimizer* of f if $\exists\ \varepsilon > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$ for which $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$.

Definition 2.1.2. A vector $\boldsymbol{x}^* \in \mathcal{X}$ is a *global minimizer* of f if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$.

In both definitions, the minimizer is said to be strict if the inequality is strict.

An optimization problem can be classified according to its type of cost function f and of set $\mathcal{X}$. An exhaustive treatment of optimization problems is out of the scope of this thesis, and some readings are here suggested for the interested reader Bertsimas and Tsitsiklis (1997); Boyd and Vandenberghe (2004); Bertsekas (2014); Horst et al. (2000). In the following we will only describe the class of optimization problems which is of interest for this work, namely unconstrained convex optimization problems.

An optimization problem is *unconstrained* if the set $\mathcal{X}$ corresponds to $\mathbb{R}^n$, that is if the search for a minimizer $\boldsymbol{x}^*$ of function f can be done in the whole space $\mathbb{R}^n$. For these problems, it is possible to give some simple conditions to help identifying whether a point $\boldsymbol{x}$ is a local minimizer, as shown in the following.

If $f \in \mathcal{C}^1$, then a necessary condition for $\boldsymbol{x}^*$ to be a (local) minimizer is for $\nabla f(\boldsymbol{x}^*)$ to be equal to 0. As a matter of fact, using the Taylor series expansion of f in $\boldsymbol{x}^*$, for all (sufficiently small) $\Delta\boldsymbol{x}$, it holds

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \nabla f(\boldsymbol{x}^*)^\top \Delta\boldsymbol{x}.$$

Since $f(\boldsymbol{x}^*)$ is a minimum, $f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*)$ has to be greater than or equal to zero, which implies that also the quantity $\nabla f(\boldsymbol{x}^*)^\top \Delta\boldsymbol{x}$ has to be non-negative. Consequently $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$.

If $f \in \mathcal{C}^2$, there is also a necessary condition involving $\nabla^2 f(\boldsymbol{x}^*)$. In particular, if $\boldsymbol{x}^*$ is a (local) minimizer, then $\nabla^2 f(\boldsymbol{x}^*)$ has to be positive semidefinite, since for the previous consideration the Taylor expansion in $\boldsymbol{x}^*$ is

$$f(\boldsymbol{x}^* + \Delta\boldsymbol{x}) - f(\boldsymbol{x}^*) \approx \Delta\boldsymbol{x}^\top \nabla^2 f(\boldsymbol{x}^*) \Delta\boldsymbol{x}, \tag{2.2}$$

and this quantity has to be greater than or equal to 0.

The previous conditions are necessary for a point $\boldsymbol{x}^*$ to be a local minimizer. To obtain a sufficient condition for a point to be a local minimizer it is necessary to enforce the Hessian to be strictly positive definite, in order to avoid saddle points. The following theorem formally states this sufficient condition for optimality:

Theorem 2.1.3. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ belong to $\mathcal{C}^2$. If vector $\boldsymbol{x}^* \in \mathbb{R}^n$ satisfies*

$$\nabla f(\boldsymbol{x}^*) = \mathbf{0}, \qquad \nabla^2 f(\boldsymbol{x}^*) \succ \mathbf{0},$$

then $\boldsymbol{x}^*$ is a (strict) local minimizer of function f .

Note that this can be easily proven since having the Hessian strictly positive definite assures that the function strictly increases in a neighborhood of $\boldsymbol{x}^*$ (see Equation (2.2)).

The other property of interest in this thesis for an optimization problem is convexity. An optimization problem is said to be *convex* if both the function f and the set $\mathcal{X}$ are convex. Note that since the set $\mathbb{R}^n$ is convex, an unconstrained optimization problem is convex if the function f is convex.

Convexity is a very nice property to have when one is dealing with an optimization problem. As a matter of fact, a convex optimization problem has a unique minimum

(even though there might be more than one minimizer); if in addition f is strictly convex, the minimizer itself is unique. These properties are here proven under the assumption that $f \in \mathcal{C}^1$ and that $\mathcal{X} = \mathbb{R}^n$, both for convenience and also because this assumption holds for most of the functions employed in the Thesis. In fact, if function f is $\mathcal{C}^1$, then f is convex if and only if

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}), \ \forall \ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \tag{2.3}$$

and it is strictly convex if and only if the previous inequality is strict for all $\boldsymbol{y} \neq \boldsymbol{x}$.

Equation (2.3) implies that if there is an $\boldsymbol{x}^* \in \mathbb{R}^n$ such that $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$, then $\boldsymbol{x}^*$ is a global minimizer. Moreover, if f is strictly convex, the strict inequality in Equation (2.3), which holds for all $yv \neq \boldsymbol{x}$, shows that $\boldsymbol{x}^*$ is the unique global minimizer of the function.

Given an unconstrained convex optimization problem, its solution is obtained in an iterative way; only if the solution has an analytical closed form (for example if f is quadratic) one can avoid an iterative approach. The algorithms developed to solve such problems start from a tentative solution of the problem and step by step move the estimate of the minimizer towards one of the real minimizers of the cost function. Next paragraph is dedicated to the description of such an algorithm, the Newton method. This algorithm will be a starting point for the applications described in Chapters 4 and 6. Newton's method needs f to be $\mathcal{C}^2$ in order to be applicable, and it also needs the Hessian $\nabla^2 f(\boldsymbol{x})$ to be positive definite at each $\boldsymbol{x} \in \mathbb{R}^n$. As so, the underlying assumption of next paragraph is that the problem to be solved is unconstrained and convex, with a twice differentiable function f with positive definite Hessian.

Before describing the algorithm, it is interesting to connect convexity and the Hessian properties. In particular, a function $f \in \mathcal{C}^2$ is convex if and only if

$$\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}, \quad \forall \ \boldsymbol{x} \in \mathbb{R}^n. \tag{2.4}$$

Differently from what happened with the gradient, strict convexity is not completely defined looking at the Hessian of f. In fact, having $\nabla^2 f(\boldsymbol{x}) \succ \mathbf{0}$ is only a sufficient but not necessary condition for the function to be strictly convex.

Newton's method

Newton's method is an optimization algorithm belonging to the class of descent methods. According to these methods, the minimizer $\boldsymbol{x}^*$ is obtained in an iterative fashion. In

particular, starting from an $\boldsymbol{x}(0) = \bar{\boldsymbol{x}}$, the update is carried out as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \varepsilon(k)\Delta\boldsymbol{x}(k), \tag{2.5}$$

where $\varepsilon(k)$ is a step size, and $\Delta\boldsymbol{x}(k)$ is the search direction. The search direction has to be chosen in order to have $f(\boldsymbol{x}(k+1)) < f(\boldsymbol{x}(k))$ as long as $\boldsymbol{x}(k)$ is not optimal. Due to (2.3), $\Delta\boldsymbol{x}(k)$ has to satisfy $\nabla f(\boldsymbol{x}(k))^\top \Delta\boldsymbol{x}(k) < 0$. According to the way the search direction is chosen, different algorithms are recovered. In particular,

- if $\Delta\boldsymbol{x}(k) = -\nabla f(\boldsymbol{x}(k))$, gradient descent is obtained
- if $\Delta\boldsymbol{x}(k) = -\nabla^2 f(\boldsymbol{x}(k))^{-1}\nabla f(\boldsymbol{x}(k))$, the Newton method is obtained

As can be inferred from Newton's search direction, to be implementable this method needs f to have a positive definite Hessian since it has to be inverted and f is convex, and so function f has to be strictly convex. Using such a direction it holds that

$$\nabla f(\boldsymbol{x}(k))^\top \Delta\boldsymbol{x}(k) = -\nabla f(\boldsymbol{x}(k))^\top \nabla^2 f(\boldsymbol{x}(k))^{-1}\nabla f(\boldsymbol{x}(k)) < 0,$$

which shows that it is in fact a descent direction if $\nabla f(\boldsymbol{x}(k))$ is not optimal. Concerning the step size $\varepsilon(k)$, this can be chosen fixed in time. If ε is chosen equal to 1, then the method is known as Pure Newton's method, while if $0 < \varepsilon < 1$ the algorithm is called Damped Newton's method.

Algorithm 2.1 contains a formal description of Damped Newton's method. The stopping condition usually regards the decrease between two consecutive steps of the algorithm. When this decrease is smaller than a threshold, than the algorithm is stopped.

Algorithm 2.1 Damped Newton's method

1: Initialize $\boldsymbol{x}(0)$ to a vector in $\mathbb{R}^n$
2: **while** stopping condition verified **do**
3: $\quad \boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \varepsilon\nabla^2 f(\boldsymbol{x}(k))^{-1}\nabla f(\boldsymbol{x}(k))$
4: **end while**

It is possible to give an interesting interpretation of the algorithm (in its pure version). In fact, second order Taylor's series expansion of f near a point $\boldsymbol{x}$ is

$$\hat{f}\,|_{\boldsymbol{x}}(\boldsymbol{x}+\boldsymbol{v}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top \boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^\top \nabla^2 f(\boldsymbol{x})\boldsymbol{v}.$$

This quadratic function is minimized choosing $\boldsymbol{v}$ equal to the Newton direction $-\nabla^2 f(\boldsymbol{x}(k))^{-1}\nabla f(\boldsymbol{x}(k))$, and so the point $\boldsymbol{x} + \boldsymbol{v}$ is the minimizer of the approximated version of f. Therefore, at each time step Newton's method quadratically approximates

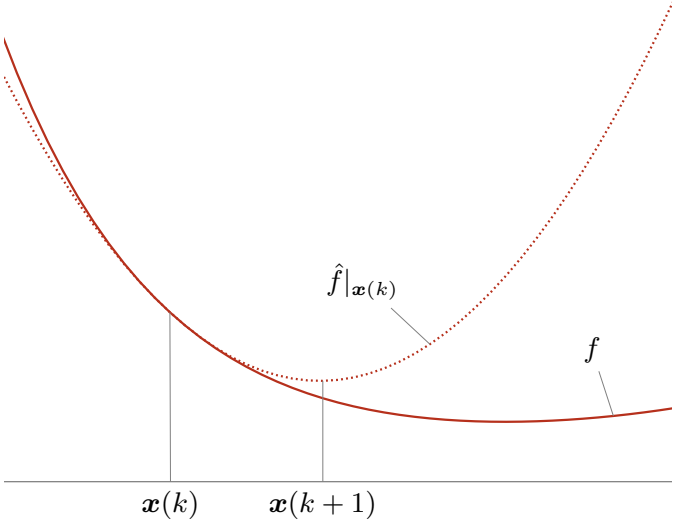


Figure 2.2: Graphical interpretation of one step of Newton's method

the function f at the current minimizer's estimate $\boldsymbol{x}(k)$, and evaluate the following estimate $\boldsymbol{x}(k+1)$ as the minimizer of this quadratic approximation of f. Figure 2.2 graphically describes how a step of the algorithm works.

A detailed proof of convergence for Newton's method can be found in Boyd and Vandenberghe (2004)[Chp. 9.5].

2.2 Distributed algorithms

In the previous section the formulation of an optimization problem was introduced in total generality, without any specific address to the set up of the system which has to solve the problem. The only algorithm presented, Newton's method, was declared to be a centralized approach without any further explanation. In this section the meaning of centralized and distributed approach is investigated, underlying advantages and disadvantages of both approaches. To clarify the terminology, the terms *computational unit*, *node* and *agent* will be used interchangeably to denote an entity which is able to perform some computation.

Given an optimization problem, a *centralized approach* relies on the presence of a unique computational node to carry out all the computation. All the information required to solve the problem is stored and processed in this node. Each step of the iterative algorithm which solves the problem consists just in the update of the estimate of the minimizer and possibly of some related variables.

On the other hand, in a *distributed approach*, the solution is obtained as a joined

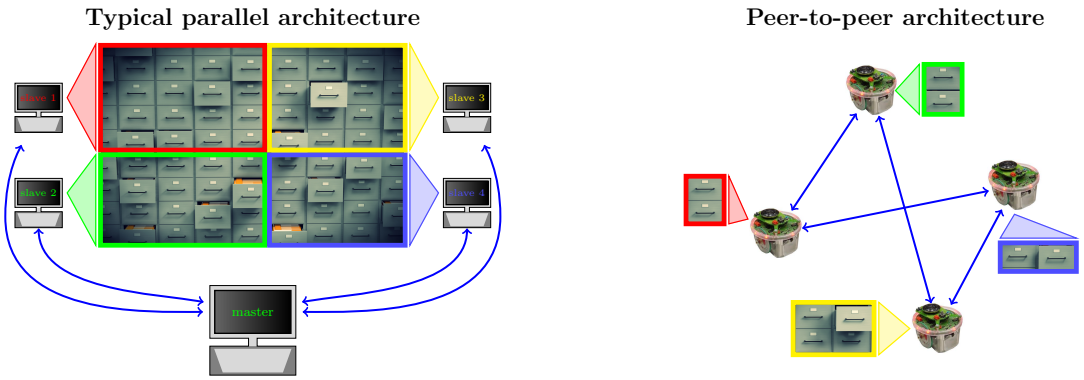


Figure 2.3: On the left a typical parallel architecture for the analysis of a big quantity of data, available to every agent. The data are divided among four slave computers, and a master computer coordinates the operations. On the right a typical peer-to-peer architecture, where each agent has its own private information and the overall analysis has to be carried out only communicating with the neighbors

Credits: the archive cabinet has been taken from http://leganerd.com/2015/09/20/big-data/

effort of a group of nodes. The situations in which a decentralized approach is used can be very different, but what is common in all of these is that each node does not have either enough information or enough computational power to solve the problem by itself. As so, the nodes have to collaborate in order to obtain the solution of the problem, and here lies the big difference with respect to the centralized approach. Collaboration requires communication between nodes, and therefore in a distributed algorithm there is always a step involving the exchange of information in addition to the "classical" update of the minimizer's estimate which is done in a centralized algorithm (see Figure 1.11).

Distributed approaches mainly evolved following two different aims: to solve very big problems and to solve problems in a peer-to-peer architecture. The research concerning the former aim is known as parallel computing, while in the following, for the ease of exposition, the research for the latter is denoted as peer-to-peer optimization. Sometimes the boundaries between these two aims are not too clear, and in many aspects the second one surmises the first one. The following description of both will show some differences between the two and also with respect to a centralized approach (Figure 2.3 shows graphically shows the differences). The discussion will also highlight the features of peer-to-peer optimization which make it adoptable in multi-agent systems.

Parallel computing aim at solving huge problems using a network of computers. All the data for the solution of the problem are (in principle) available at a unique computer, but their elaboration in a unique computational unit is too complex. The idea is then to divide the effort for the solution of the problem among different computers, as opposed to solving the whole problem using just one unit Tsitsiklis (1984); Bertsekas and

Tsitsiklis (1989). As long as the size of the problem is not too big, one can employ either a centralized or a distributed approach. A centralized approach might be preferable since there exist many efficient centralized algorithms for the solution of an optimization problem and there is no need of information's exchange. However, as the size of the problem to be solved becomes larger and larger, a centralized approach becomes more difficult. As a matter of fact it requires a unique computational unit sized for the problem, that is with huge computational and memory capabilities, making this solution very expensive. Moreover, not only the node has to be properly sized, but also the management of the big quantity of data becomes difficult from an algorithmic point of view. Differently, if a distributed approach is employed, the size of the problem at each node is smaller, and so easier to manage and requiring much cheaper nodes. The advent of the Big Data era, with an enormous quantity of data collected every day (for example from scientific experiments or from the Internet usage) gave a big boost to research in distributed algorithms in order to achieve parallel computing Dobre and Xhafa (2014); Tsai, Lin, and Ke (2016); Boyd et al. (2011). However, in parallel computing the choice of solving the problem in a distributed way is done on purpose, it is not a strict consequence of the problem, since in principle this can be solved in a centralized way. The architectures employed for parallel computing are specifically designed for collaboration among nodes. Therefore, even though the algorithms require communication, the communication set up is usually reliable and synchronism of the nodes is not so difficult to achieve. Another very important feature is that the architectures used in parallel computing are usually composed by a master node and a lot of slave nodes, where the slave nodes solve a part of the problem while the master merges the different solutions.

As opposed to parallel computing, in a peer-to-peer architecture there is no hierarchy among the agents involved in the computation, there is just a group of similar agents which need to solve a problem in a collaborative manner, with no distinction of importance among them. The agents themselves might not be computers, but other entities endowed with some computational power (like sensors or robots). This is the set-up of interest for multi-agent systems since they are made of generic smart agents among which there might be (in total generality) no hierarchy. Examples are systems which cover huge geographical areas (and where cabling is not possible) or system where a presence of a central entity is not possible or advisable. Imagine a wireless sensor network with sensors placed on a very wide area. In this case it is infeasible to have each sensor communicate to a central unit and usually there are only few nodes which can communicate to some higher level entity. As a consequence, the sensors first have to collaborate among themselves to perform some data fusion, and then the information is sent to the high level entity. Furthermore,

in parallel computing the problem usually allows a (perhaps difficult) centralized solution, while in peer-to-peer optimization the interest is also for problems that can only be solved using a distributed solution due to their nature. In particular, the information known by each peer might be private and so cannot be collected at a central node. In a multi-agent set-up this is not uncommon. For example it is possible to have a group of agents, each endowed with a private cost function which depends on the same variable for all the agents. If these agents want to agree on the choice of this variable in order to minimize the sum of the private cost functions, it would be impossible to solve the problem in a centralized way, since no agent would be willing to disclose its private function to another entity. In this case only a distributed approach is possible. In general, every time there is a privacy issue, a distributed approach is preferable to a centralized one.

Aside from the cases in which it is necessary, in a multi-agent system a distributed approach (as intended in peer-to-peer optimization) has some advantages with respect to a centralized solution. The first one is that since all the agents in the system have computational power, it would be wasteful not to exploit it. Another advantage is that the system does not rely on a central unit for all the computation, and this makes the approach more robust. As a matter of fact, if the central unit fails no agent in the system can act, while if a (properly developed) distributed algorithm is employed, apart from some potential failing agents, the other can continue to act. However, the necessary communication step between the agents can be really challenging. Considering peer-to-peer optimization for a multi-agent set-up, communication is generally unreliable. Wireless communication, with its wide applicability and usage, is an example of unreliable transmission mean and is adopted for example in sensor networks or in groups of mobile robots. Moreover, when peer-to-peer optimization is applied to multi-agent systems, most of the time it is not possible to decide the communication network (that is which agent communicates with which). Using again the wireless communication as an example, an agent can communicate only with those that are near to it, and so the communication network depends on the agents' disposition. Chapter 5 will show that the communication network is a determinant factor for the convergence rate (intuitively how many iterations are necessary to get a good approximation of the optimal solution).

The state of the art for algorithms for peer-to-peer architectures will be explored in detail later on in the thesis. Depending on the specific problem analyzed, the kind of algorithms employed to solve it are different. Therefore, it is more straightforward to explore the literature connecting it to the specific problem.

Before describing the challenges of real world communication, the communication network for the multi-agent systems examined in this thesis are here formally described.

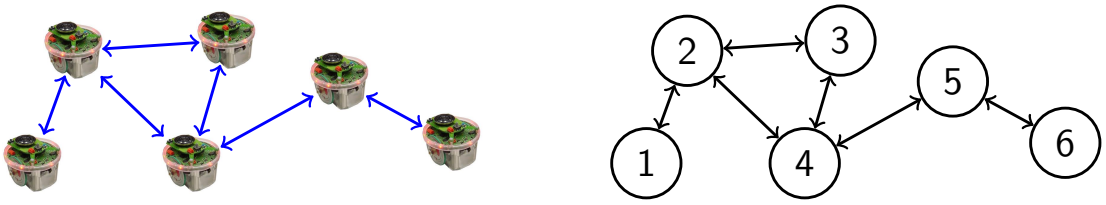


Figure 2.4: On the left a multi-agent system composed by $N = 6$ robots. As always, a blue arrow between two agents shows that these agents can communicate. The right part of the figure shows the underlying communication graph.

All the multi-agent systems analyzed are composed of N agents. These agents can communicate according to an underlying communication graph $\mathcal{G}$. This (directed) graph $\mathcal{G}$ is composed by N nodes, one for each agent in the multi-agent system, and an element $(i, j),\ i, j \in \{1, \ldots, N\}$, exists in its edge set $\mathcal{E}$ if and only if agent i can communicate with agent j. Figure 2.4 shows an example of multi-agent system with its underlying communication graph. In this dissertation, the communication network is taken to be fixed in time, but there exist distributed algorithms that can work with a time-varying communication graph. Apart from time invariance, another fundamental assumption throughout the dissertation is the following

Assumption 2.2.1 ((Strongly) Connected Graph)**.** The directed (undirected) communication graph $\mathcal{G}$ is strongly connected (connected respectively).

This assumption in a way implies that the information can be circulated among all the agents in the network. Communication is (clearly) allowed only between one-hop neighbors in $\mathcal{G}$. This will be a limiting factor for the development of some of the algorithms (see in particular Chapter 4).

2.3 Communication issues

Links' reliability

Communication is one of the fundamental differences between centralized and distributed methods, as already pointed out in the introduction. In the majority of distributed algorithms, it is assumed that the information's exchange is a reliable step, that is every packet sent by an agent to one of its neighbors is received for sure by the neighbor. However, this is not usually the case in a real communication system. In multi-agent

systems, where wireless communication is often adopted, the assumption of reliable communication is rarely met. As a matter of fact, wireless communication is prone to packet losses due to disturbances (especially if the power employed for communication is limited) Hou and Kumar (2012); Zamalloa and Krishnamachari (2007). In any case, also for other kinds of communication it is difficult to achieve perfect communication and some packets can be lost due to ambient noise, collisions, or other effects (for example, during a congestion period, also the Internet can lose some packets). When a distributed algorithm is implemented in a real application, it is therefore necessary to be able to deal with packet losses. As a matter of fact, if the presence of packet losses is ignored, when an agent does not receive the packet it is expecting, the agent does not know how to act (see Figure 1.13). There are two possible solution to adopt in order to tackle the consequences of packet losses: either modifying the communication step or modifying the algorithm.

A very simple way to deal with packet losses is modifying the communication protocol. In particular, the use of acknowledge messages can easily solve the problem of unreliable communication. Without acknowledge, from the point of view of the sender, a communication exchange is concluded upon sending the message. On the other hand, using a protocol with acknowledge, the receiver, upon reception of the message, sends a message, called acknowledge message or ACK, to the sender. Only the reception of the ACK by the sender concludes the exchange, while if the sender does not receive the ACK sent from a given agent within a (fixed) period of time, it sends the packet again, and repeats the sending until reception of the ACK (see Figure 1.14). Clearly, with such a communication protocol, packet losses are avoided since the communication step goes on until all the packets' exchanges are successful. However, the use of ACKs has its own drawbacks. One of these is that it requires the receiver to send a message; in scenarios where energy is limited, for example in WSNs, the consumption of energy is reduced to the minimum. Since communication is a very demanding operation from a power consumption point of view, in these scenarios this additional packets' exchange should be avoided. Another drawback is that the communication step does not have a fixed length, since when a packet (or the related ACK, even though this second event rarely happens) is lost, the sender after some time has to send the message again and this can happen for many consecutive times if the communication link is particularly disturbed. Obviously, a longer communication step implies longer algorithm's execution time, and if the communication system is particularly unreliable the duration of the communication step can become considerable. Also, even if the communication is reliable, the sender has to wait for all its neighbors' ACKs, and since it can handle just one message at a time,

the time interval for communication is longer than the one required if no ACK is used.

Every time a distributed algorithm has a fixed communication graph and does not consider packet losses, an underlying assumption of perfect reliability of the communication network is made, or the use of an acknowledge protocol is tacit. Note that one might think that assuming a time-varying communication network implies that one is somehow also dealing with packet losses. In Chapters 5 and 6, it is shown that this is not the case. As a matter of fact, in many cases the analyses of scenarios with time varying communication graphs assume that at each time step the agents exactly know to whom it is sending the packet (and so packet losses are not considered).

Alternatively to the use of ACKs, one can deal with unreliable communications modifying the algorithm itself. In this case, the idea is to develop an algorithm in which the agents send the packets but do not care whether the neighbors actually receive the message or not. This kind of algorithms are hereafter denoted as *robust* (to packet losses) and among the scopes of this thesis is the introduction of robust algorithms which rely on a communication protocol without acknowledge. There are different possible ways to make an algorithm robust, and in this dissertation three different methods are analyzed. In particular, one of the solutions presented relies on a modification of the (non robust) algorithm (recall Figure 1.18), another one on the use of memory (see Figure 1.20) and still another on the use of additional variables and a consequent algorithm modification (remember Figure 1.22).

Using a robust algorithm makes the communication protocol easier to implement (with respect to a protocol that uses ACKs), solves the problem of the variable duration of the communication step, making it shorter, and also avoids non strictly necessary communication. However, the convergence proof becomes more complicated, since the set of information used by each agent is possibly different among the agents, and this introduces some difficulties.

In the following chapters, when the communication network is unreliable, the following assumption holds:

Assumption 2.3.1 (Bounded packet losses)**.** There exists a positive integer h such that the number of consecutive communication failures over every directed edge in the communication graph is smaller than h.

According to this assumption, a link in the communication network cannot be unreliable for an infinite number of consecutive times.

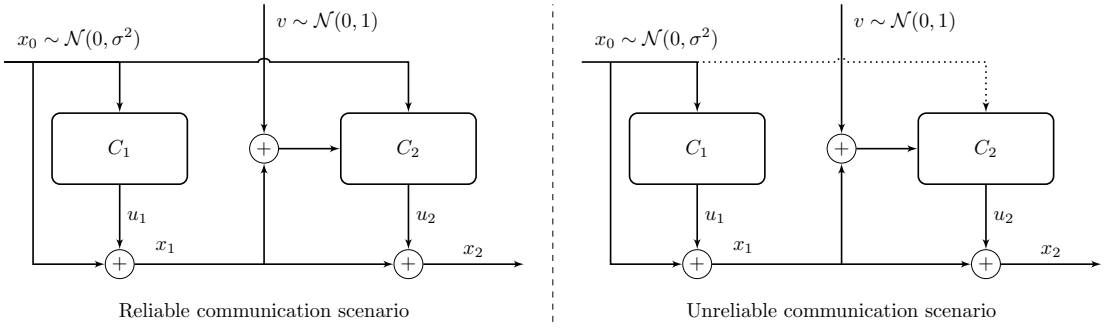


Figure 2.5: System employed in the Witsenhausen counterexample. On the left side the system with reliable communication, on the right the system with unreliable communication (a dashed line indicate an unreliable link).

Before analyzing the challenges of synchronism, it might be useful to show how important the assumption on the reliability of communication is. As a matter of fact, one might think that the loss of packets might not be too detrimental for the overall system. The following discussion, based on Witsenhausen's counterexample Witsenhausen (1968); Sahai and Grover (2010), shows the effects of an unreliable communication system.

Witsenhausen's counterexample

The deceivingly simple example introduced by Witsenhausen in 1968 Witsenhausen (1968) aimed at showing the difficulties introduced by a decentralized control. It is described here in a slightly different context in order to show the importance of a perfect communication assumption.

Consider the system on the left part of Figure 2.5. It is a system where a scalar quantity (x_0) evolves in discrete time according to linear dynamics. Two controllers act on the state in two consecutive steps. Controller C_1 receives as input x_0 and can act on the state (x_0 again), adding u_1, which is a function of the controller's input x_0. A noisy version of the resulting state, x_1, is given as input to the second controller, C_2, together with the variable x_0. C_2 then acts on the state x_1 adding u_2, generating the resulting variable x_2. The inputs of the overall system, that is x_0 and v (the noise measurement for x_1), are Gaussian variables, respectively $x_0 \sim \mathcal{N}(0, \sigma^2)$, $\sigma > 0$, and $v \sim \mathcal{N}(0, 1)$. The aim of the overall system is to solve an optimal control problem. The cost function to be minimized is the expectation of the following quadratic function

$$\mathbb{E}\left[ku_1^2 + x_2^2\right], \ k > 0$$

and the minimization is done on the controllers' functions u_1 and u_2. In this first system,

perfect communication is assumed, and the overall problem can be interpreted as a Linear Quadratic Gaussian (LQG) one. An LQG problem can be optimally solved choosing the controllers' functions as linear functions of the inputs using Riccati's equations Anderson and Moore (1971). For the system analyzed here, the optimal choices for the controllers' functions simply are

$$u_1(x_0) = 0, \quad u_2(x_0, x_1 + v) = -x_0,$$

according to which it is easy to verify that the value of the cost function is 0.

Now, consider the system on the right of Figure 2.5. In this new system, the communication link which brings x_0 to C_2 is unreliable. Assume also that the controller C_1 does not know whether C_2 receives or not x_0 (that is there are no ACKs). The previous choice of functions u_1 and u_2 is not possible anymore, since when C_2 does not receive x_0 the controller would not know what to do. The new control strategy to be developed has therefore to ignore the fact that C_2 may receive x_0 as input. This new scenario is the one introduced by Witsenhausen, who wanted to show that, even though this problem seems very similar to an LQG problem again, the fact of not sharing the same variable can make the solution very challenging (and so that distributed solutions can be much more arduous). Witsenhausen was in fact able to show that if controller C_1 is constrained to be linear (that is u_1 is chosen as a linear function of x_0) than also controller C_2 has to be a linear fnction of $x_1 + v$ to minimize the cost function. At that time, this solution was conjectured to be the optimal one even without the restriction on C_1, due to its similarity to the standard LQG problem Witsenhausen (1968). However, Witsenhausen showed the existence (for some values of k and σ) of non-linear functions $\tilde{u}_1$ and $\tilde{u}_2$ which achieve a smaller value of the cost function than the one obtained using linear controllers. Interestingly, the non-linear functions proposed by Witsenhausen have a "communication" interpretation. As a matter of fact, the function $\tilde{u}_1$ is chosen in such a way that the true value of the resulting variable x_1 can be with good probability recovered by C_2 even though this controller receives only a noisy observation of this variable. Somehow, the control acts as a communication system, and this kind of implicit communication is known in the control literature as *signaling*. Since Witsenhausen was also able to prove the existence of an optimal solution, his counterexample showed that linear controllers are not the best choice in a decentralized scenario. Until now, the optimal solution of this "trivial" example (which is a non-convex and NP-complete problem) is unknown, and only some numerical solutions, which are thought to be good approximation of the true solution, exist.

For the sake of this thesis, this counterexample showed how much the solution of a problem can be different if one considers reliable or unreliable communication.

Synchronism

A distributed algorithm can be classified either as synchronous or as asynchronous Kung (1976). In the specific case of optimization with a peer-to-peer architecture, a distributed algorithm is *synchronous* if all the agents in the system need to exchange information at the same time, while it is *asynchronous* if the communication step can be performed only by a subset of the nodes (and not necessarily all of them at the same time). In particular, in a synchronous algorithm an iteration of the algorithm involves all the nodes in the system, while in an asynchronous one an iteration can modify only the variables of a subset of the nodes. In accordance to the algorithm, the former employs a synchronous communication protocol, while the latter an asynchronous one (recall Figure 1.15).

In a real situation, performing the communication step for a synchronous algorithm can be challenging. As a matter of fact, the nodes have to be able to verify whether all the other nodes in the network have reached the communication step before performing it, or there has to be a common notion of time among the nodes (and some guarantee on the duration of the update step). As the number of agents in the system increases, verifying synchronization or maintaining a common notion of time becomes more and more challenging from a technological point of view. Moreover, this type of communication protocol can slow down the entire algorithm. Suppose in fact that there exists in the system a node that is slower than the other ones. Since all the nodes have to synchronize before performing the message exchange, at each iteration all the nodes have to wait for the slowest one.

Asynchronous algorithms, on the other hand, require much less coordination if compared to synchronous ones. As a matter of fact, only a small subset of all the nodes in the network performs the communication and updating steps. In order to clarify the terminology, the verbs *to wake up*, *to be activated* or *to be selected* are referred to the nodes/edges that at each iteration are the ones that determine which agents exchange information. In the following some of the most used asynchronous protocols are described. In the *asymmetric broadcast* protocol, at each iteration there is only one node transmitting information to its out-neighbours, which, based on the received messages, update their internal variables. The *convergecast* (or *coordinate broadcast*) can be considered as the dual protocol of the broadcast asymmetric. Indeed, at each iteration, there is only one node which wakes up, but, instead of sending information, it polls all its in-neighbours in order to receive from them some desired messages. In *asymmetric gossip* again only one node wakes up but it sends information to only one of its out-neighbours, typically randomly chosen. Finally, the *symmetric gossip* is a protocol that requires bidirectional communication, that is the communication graph $\mathcal{G}$ has to be

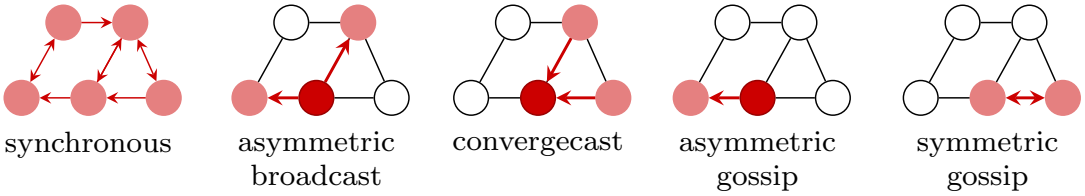


Figure 2.6: Synchronous and asynchronous communication protocols. The communication graph is the same for all the protocols except the last one (since symmetric gossip requires an undirected graph), but in the asynchronous case the direction of the non-active edges is not shown to help readability. For the asymmetric and coordinate broadcast and for the asymmetric gossip, the opaque red node is the one activated for the current iteration.

undirected; during each iteration an edge of the graph is selected and only the two nodes which are pointed by this edge exchange information with each other. Figure 2.6 gives a pictorial description of the synchronous and asynchronous protocols just described. Asynchronous algorithms do not require a common notion of time, and so are easier to implement when the number of agents becomes bigger and bigger. In addition, using an asynchronous protocol, a potential slower node in the network slows down the iteration only when it is activated, so when this node is not involved in an iteration, the latter is faster. However, an asynchronous algorithm will probably require more iterations than a synchronous counterpart, since at each iteration only a subset of nodes makes some computation.

The algorithms presented in this thesis employ asynchronous protocols, and in particular asymmetric broadcast (sometimes slightly modified), which represents a widely used communication protocol in wireless sensor networks applications. As already described, according to this protocol, at each iteration a single node wakes up and exchange information with its neighbors. The following assumption on the activation's frequency will be used in all the application examined in the manuscript.

Assumption 2.3.2 (Persistent agents' activation)**.** There exists a positive integer number τ such that each agent in the system wakes up at least once within the interval $[k, k+\tau)$, for all $k \in \mathbb{N}$.

From a practical point of view, it is possible to imagine that each node has an internal clock, and the time interval between two consecutive activation is equal to a realization of a random exponential variable.

2.4 Applications' features that call for a distributed approach

This section wants to underline why one should choose a distributed approach in the applications analyzed in this thesis. In particular, the salient features of each problem that suggest a distributed algorithm are analyzed. The idea is also to show that one can decide to employ a distributed algorithm to answer to different demands.

Patrolling problem

In the patrolling problem, analyzed in Chapter 3, the presence of a central unit is not considered, and so the distributed approach to solve the problem is the only possible solution (and the same fact holds for all the other application presented). However, also the problem structure makes a distributed approach desirable. As a matter of fact, according to its formulation as an optimization problem, the optimization variable can be divided into pieces, one for each agent (that is each camera) of the network. In order to patrol the area, each agent only needs to know the part of the optimization variable which strictly belong to itself, and this suggests that a distributed approach is adequate, since each camera is only interested in its own part of the optimization variable. In a multi-agent set-up, the problems in which each agent in the network is only interested in a piece of the optimization variable can be denoted as ***local estimation*** problems (sometime referred as *partition based* problem). If a distributed approach is employed, each agent can work only on its part of the optimization variable (and the information's exchange among neighbors assures that the overall optimization variable converges to the minimizer). In this way, even though the number of agents increases in the network, the algorithm run by each agent remains simple. Therefore, this distributed algorithm scales well with the system's dimension. Moreover, as will be better explained in Chapter 3, a distributed approach in this case is also desirable, since the patrolling problem is strictly related to the tracking problem. Namely, if an intruder is found during the patrolling, the camera which finds it leaves the patrolling mode and tracks the intruder. At the same time, its neighboring cameras have to start patrolling the area which is not patrolled anymore. A distributed approach helps to make such an arrangement in a very fast way. In case of a centralized approach, this might require more time.

Locally coupled cost

The second problem analyzed is the minimization of the sum of local cost functions, which are locally coupled. Chapter 4 is dedicated to this problem. As can be inferred looking

at Figure 1.19, each agent needs only information from its neighbors to compute its cost function. It is also possible to foresee that in order to minimize the sum of the cost functions, an agent is interested in looking only at a partial part of the overall network. The solution here proposed, in fact, is inspired by the centralized Newton method, and so each agent needs to evaluate a part of the overall gradient and Hessian. The particular form of the problem generally implies that this part of the gradient and of the Hessian depends on information coming not only from its one-hop neighbors but also from its two-hops neighbors. Due to these local aspects, it makes sense to apply a distributed algorithm. Moreover, in many cases, each agent is only interested in its own part of the (overall) optimization variable, and so this kind of problem can be also seen as a local estimation problem, as for the patrolling problem.

It is important to note that the dependence on two-hops neighbors' information is problematic in the communication set-up of interest for the thesis, described at the end of Section 2.2. As a matter of fact, direct communication between two-hops neighbors is not permitted, and to make the algorithm implementable it is necessary to let each agent remember the last packet received by the one-hop neighbors. Interestingly, as will be proved in Chapter 4, this additional memory, apart from solving the two-hops communication issue, also solves the problem of packet losses.

Consensus problem and quadratic cost minimization

Concerning the consensus problem, examined in Chapter 5, a distributed approach is advisable because in this case each agent in the system has its own quantity, so the information is local by nature. Moreover, if some privacy issues exists, the use of a distributed approach helps to preserve it (it the algorithm is properly designed). Considering the problem as the minimization of the sum of quadratic cost functions (a function for each agent), conversely to the previous applications, in this case each agent in the system is interested in retrieving the full minimizer of the overall cost function (and not just part of it). Due to this feature, consensus is a ***global estimation*** problem.

Additively separable cost function

The last application examined in this dissertation involves the minimization of additively separable cost functions. In a multi-agent set-up such a problem exists if each agent is endowed with a private cost function and each agent wants to minimize the sum of the cost functions. As a consequence, a distributed approach is necessary. As already pointed out in the introduction, the consensus problem is a special case of additively separable cost function minimization, and, as so, the problem analyzed in Chapter 6 is a

global estimation one. Interestingly, precisely because it is a global estimation problem, consensus can play an important role in the development of a distributed algorithm for the solution of the problem. This intuition is confirmed by some of the existing algorithms to solve this kind of problems.

3

Patrolling for camera networks

The contents of this chapter are based on the paper

Bof N., Carli R., Cenedese A., and Schenato L. Asynchronous distributed camera network patrolling under unreliable communication. *IEEE Transactions on Automatic Control*, 62(11):5982–5989, 2017b.

This chapter is devoted to the study of the patrolling problem for smart camera networks Aghajan and Cavallaro (2009). Particular attention is therefore given to features related to this multi-agent system. However, the results obtained can be useful also for robot networks where the agents have to monitor an area Acevedo, Arrue, Maza, and Ollero (2013).

In the first part of the chapter the patrolling problem is introduced, together with an optimal formulation for the same. A preliminary algorithm to solve the optimization problem is then described. This algorithm is asynchronous but relies on a reliable communication system and it is possible to show its convergence to a unique point in the minimizers' set. On the other hand, when the communication is not reliable, this first algorithm is not usable anymore, because part of the area to be monitored might remain temporarily uncovered and this is not admissible. An adjustment of the algorithm is then proposed in order to deal with packet losses, and its convergence is showed. However, in this case it is only possible to show convergence to the set of minimizers.

3.1 Introduction and state of the art

Video surveillance systems are nowadays increasingly used for security and prevention purposes in a variety of different situations. They can be used as a deterrent for intruders, but also for the early detection of anomalous events.

Important tasks required to these systems are target acquisition, tracking, activity recognition Song, Kamal, Soto, Ding, Farrell, and Roy-Chowdhury (2010); Kariotoglou, Raimondo, Summers, and Lygeros (2015); Ding, Song, Morye, Farrell, and Roy-Chowdhury (2012) and patrolling.

Here, the task analyzed is the patrolling problem for networks of Pan-Tilt-Zoom (PTZ) cameras. This problem corresponds to the repetitive monitoring of a perimeter or of an area realized by a group of cameras, in order to be able to detect intruders or to locate unexpected events. The examined scenario is given by a group of already deployed and fixed PTZ cameras that have to patrol a given one-dimensional environment.

The patrolling problem on a one-dimensional environment using a network of PTZ cameras is studied in Alberton, Carli, Cenedese, and Schenato (2012), where it is reduced to a partitioning problem. This approach is effective in case the intruder is static. To deal with dynamic intruders, the partitioning has to be combined with a given schedule for the movements of the cameras, as shown in Pasqualetti, Zanella, Peters, Spindler, Carli, and Bullo (2014); Borra, Pasqualetti, and Bullo (2015). For the patrolling of two-dimensional areas, randomized strategies have been proposed in Huck, Kariotoglou, Summers, Raimondo, and Lygeros (2012); Raimondo, Kariotoglou, Summers, and Lygeros (2011).

The patrolling problem for networks of PTZ cameras has similarities to that for mobile-agents, which is studied in many papers. With no intention of providing an exhaustive overview on the subject, some related literature is reported in the following. The problem of patrolling different disjoint areas using agents that can move from one area to the other is studied for example in Chevaleyre (2004) and Chu, Glad, Simonin, Sempé, Drogoul, and Charpillet (2007): the first solves it as a travelling salesman problem, while the latter uses a swarm intelligence approach. Different solutions are provided in Mao and Ray (2014), where reinforcement learning is adopted to deal with a similar problem and in Cassandras, Lin, and Ding (2013), where the patrolling of a one-dimensional environment is addressed solving an optimal control problem. More interestingly with respect to the solution proposed in this chapter, the authors of Acevedo et al. (2013) and Acevedo, Arrue, Diaz-Bañez, Ventura, Maza, and Ollero (2014) consider the patrolling of a one or two-dimensional environment and reduce this problem to a partitioning one similar to that in Alberton et al. (2012).

Like the algorithms that will be proposed in the remaining of this chapter, many of the articles just cited adopt a distributed approach to solve the patrolling problem. For the particular multi-agent system which is here analyzed, it is possible to mention some specific advantages in using such an approach. As will be shown later, a distributed approach is really scalable in this case, and require information only from neighboring agents. Moreover, it may be difficult to collect all the information in a single unit if the communication is not reliable. A distributed approach may also be safer in presence of attackers/intruders, who would have to compromise each single camera and not just a central unit. Finally a distributed algorithm can adapt fast to dynamic scenarios in which cameras switch from patrolling mode to tracking mode and vice-versa, or in which some cameras may be malfunctioning.

All the distributed algorithms proposed in the literature for the patrolling problem assume reliable communications. If this is realistic enough for cabled camera networks, this assumption may be inaccurate when the camera networks are wireless. These networks are becoming very popular thanks to their reduced installation and configuration costs and increased bandwidth performances. The aim of this chapter is to propose an asynchronous distributed algorithm for camera network patrolling that is guaranteed to converge to an optimal solution, while ensuring certain coverage properties even if the communication is not reliable and there is no acknowledges' exchange.

3.2 Problem formulation

The problem which is specifically addressed is the patrolling of a one-dimensional environment of finite length using a finite number of cameras. This situation is typical of outdoor camera networks monitoring the boundary of an area of interest, such as urban neighbourhoods or large facility perimeters. Let $\mathcal{L} = [0, L]$, $L > 0$, denote the segment to be monitored and let N be the cardinality of the cameras' set, with the cameras labeled 1 through N. For the sake of simplicity, it is assumed that (a) the cameras are 1-d.o.f., meaning that the field of view (f.o.v.) of each camera is allowed to change due to pan movements only, (b) the cameras have fixed coverage range, meaning that during pan movements the camera coverage range is not altered by the view perspective, (c) cameras have point f.o.v..

The *patrolling range* D_i is defined as the total allowed area that the i-th camera can patrol due to the scenario topology, the agent configuration and its physical constraints. More formally

$$D_i = \left[\underline{d}_i, \overline{d}_i\right] \subset \mathcal{L}, \qquad \underline{d}_i < \overline{d}_i,$$

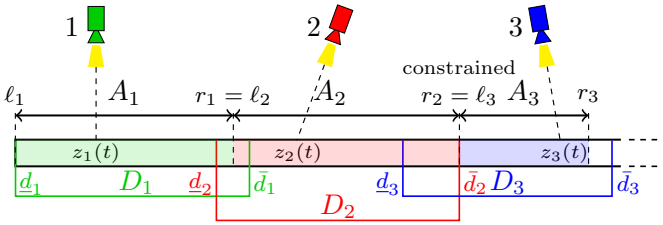


Figure 3.1: Portion of the perimeter to be patrolled. The figure shows the physical coverages $\{D_i\}$ and the patrolling areas $\{A_i\}$ for the first three cameras of the surveillance system.

where $\underline{d}_i$, $\overline{d}_i$ are the left and the right extremes of the interval D_i, respectively. In order to be able to perform an effective monitoring, the cameras have to be disposed in such a way that the patrolling ranges D_i, $i \in \{1, \dots, N\}$ satisfy the following *interlacing* physical coverage constraints,

$$\underline{d}_i \leq \underline{d}_{i+1} \leq \overline{d}_i \leq \overline{d}_{i+1}, \quad i = 1, \dots, N-1, \tag{3.1}$$

together with $\underline{d}_1 = 0$ and $\overline{d}_N = L$. According to these conditions, in fact, $\cup_{i=1}^{N} D_i = \mathcal{L}$, and so the segment $\mathcal{L}$ can be fully patrolled. This aspect is further discussed in Belgioioso, Cenedese, and Michieletto (2016).

The *max speed* $\overline{v}_i \in \mathbb{R}^+$ is the maximum speed of the i-th camera during pan movements, i.e., $|v_i(k)| \leq \overline{v}_i$.

The *camera position* $z_i(t)$ is the position of the f.o.v. of the i-th camera as a function of the time variable t.

The *patrolling area* $A_i = [\ell_i, r_i]$, with $\ell_i \leq r_i$, respectively the left and right extremes of A_i, denotes the area that is actively patrolled by the i-th camera and, differently from D_i, can be updated at discrete times $k > 0$, namely $A_i = A_i(k)$. The **physical constraints** $A_i \subseteq D_i$ must hold for all cameras at any time.

Note that the patrolling areas $\{A_i\}_{i=1}^{N}$ can be equivalently described by the pair of vectors $\boldsymbol{r} = [r_1 \ \dots \ r_n]$ and $\boldsymbol{\ell} = [\ell_1 \ \dots \ \ell_N] \in \mathbb{R}^N$, or by vector $\boldsymbol{\xi} = [\boldsymbol{r}^\top \ \boldsymbol{\ell}^\top]^\top \in \mathbb{R}^{2N}$.

Figure 3.1 depicts a portion of the surveillance scenario.

In order to define the patrolling problem as an optimization problem, it is necessary to introduce a suitable cost function and, accordingly, state an optimality criterion. After introducing an initial optimization problem, whose solution would give the best possible patrolling scheduling, this will be modified in order to have a different but much simpler one. The solution of this new problem are sometime suboptimal with respect to the former one, but what is truly gained solving the initial optimization problem is a very small benefit which perhaps does not justify the much higher complexity of the solution. For $x \in \mathcal{L}$ and $j \in \mathbb{N}$, let $\bar{\tau}_j = [\underline{t}_j(x), \bar{t}_j(x)]$ be the time interval during which the point

x is visited for the $j-$th time (counting times starting from time $t = 0$) by at least a camera $i \in \{1, \dots, N\}$. Namely, for all $t \in \bar{\tau}_j$ there exists a camera i such that $z_i(k) = x$. If point x is visited at time k only by a passing camera i, i.e, $z_i(k) = x$ and $\dot{z}_i(k) \neq 0$, it holds $\underline{t}_j(x) = \bar{t}_j(x)$. Now, for each point $x \in \mathcal{L}$, it is possible to introduce the following cost function

$$\gamma(x) := \begin{cases} \sup\limits_{j \in \mathbb{N}} \left(\underline{t}_{j+1}(x) - \bar{t}_j(x) \right) & \text{if } \forall\, j \quad \exists\, \underline{t}_{j+1}(x) < \infty \\ +\infty & \text{otherwise.} \end{cases}$$

The global cost function is then given by the following time lag

$$T_{lag} = \sup_{x \in \mathcal{L}} \gamma(x)$$

and the corresponding problem is the minimization of T_{lag}, that is the minimization of the elapsed time between two consecutive visits of the same location of $\mathcal{L}$. To have $T_{lag} < \infty$ it is necessary that each point $x \in \mathcal{L}$ belongs to at least one patrolling area A_i, namely, that the **covering constraint** $\bigcup_{i \in \{1,\dots,N\}} A_i = \mathcal{L}$ is satisfied. Observe that, if the following **interlacing constraints**

$$\ell_i < \ell_{i+1} \leq r_i < r_{i+1}, \ \forall i = 1, \dots, N \tag{3.2}$$

are satisfied and if it also holds that $\ell_1 = 0$ and $r_N = L$, then the covering constraint is satisfied. In the following, the standing assumption is that the conditions $\ell_1 = 0$ and $r_N = L$ are always satisfied.

The minimization of T_{lag} in case there are no physical constraints for the cameras is a tricky problem. As a matter of fact, one could reasonably think to use a partitioning approach to solve the problem as done in Czyzowicz, Gasieniec, Kosowski, and Kranakis (2011). In the aforementioned paper, the following conjecture is given:

Conjecture 3.2.1. Assume $D_i = \mathcal{L}$ for all i. Then the optimal minimum value for T_{lag} is attained by partitioning $\mathcal{L}$ into non-overlapping intervals of lengths proportional to the cameras speeds, specifically

$$\ell_1 = 0, \qquad r_N = L, \qquad r_i = \ell_{i+1} = \ell_i + \frac{\bar{v}_i}{\sum_{i=1}^{N} \bar{v}_i} L,$$

and letting each camera i sweeping back and forth A_i at its maximum pan speed $\bar{v}_i$. This strategy obtains

$$T_{lag} = \frac{2L}{\sum_{i=1}^{N} \bar{v}_i}. \tag{3.3}$$

The above conjecture has been shown to be true in the following two scenarios Kawamura and Kobayashi (2015):

1. when $N = 1, 2, 3$;
2. for any $N > 3$ in case $\bar{v}_1, \dots \bar{v}_N$ are all equal to each other (i.e, there exist $\bar{v}$ such that $\bar{v}_1 = \cdots = \bar{v}_N = \bar{v}$).

Remarkably, in case the maximum pan speeds are not all equal to each other, the authors in Kawamura and Kobayashi (2015) have shown the existence of some particular N-tuples $(\bar{v}_1, \dots, \bar{v}_N)$, $N > 3$, for which it is possible to design cameras' trajectories attaining a value of T_{lag} smaller than that in (3.3), thus invalidating Conjecture 3.2.1. Despite the presence of these counterexamples, in Kawamura and Kobayashi (2015) it is however argued that the solution illustrated in Czyzowicz et al. (2011) attains a value of T_{lag} that is very close to the optimal one, that is, it can be regarded as a significant sub-optimal solution.

Therefore, since the very simple *"partitioning and sweeping back and forth at maximum speed"* strategy described in Conjecture 3.2.1 is likely to be an almost optimal solution of the patrolling problem, one can change the set of possible cameras' trajectories: instead of minimizing the patrolling time lag among all possible trajectories, the set of possible trajectories is restricted, hoping that the solution on the restricted will not be too far from optimality.

In particular, the possible trajectories are constructed in the following way: the environment is partitioned into N parts, and each camera sweep its own part of the perimeter at maximum speed. The problem is then to find the partition that minimizes T_{lag} under this restriction on the trajectories. Therefore, the problem that is actually tackled is a partitioning one. Apart from its semplicity and suboptimality, the choice of partitioning acquires even more significance in the set-up considered here, since the presence of physical constraints might impose severe limitations to the areas to be patrolled by the cameras.

After partitioning, by sweeping back and forth at speed $\bar{v}_i$ a given interval $A_i = [\ell_i, r_i]$, the time lag for camera i, $i = 1, \dots, N$, is $T_{lag}(A_i) := \frac{2|A_i|}{\bar{v}_i}$, where $|A_i| := r_i - \ell_i$. As a consequence, the problem to be solved can be cast as

$$\mathcal{P}_1 : \quad T^*_{\mathcal{P}_1} = \min_{A_1,\dots,A_N} \max_i \{T_{lag}(A_i)\}$$
$$s.t. \begin{cases} A_i \subseteq D_i, & i = 1, \dots, N \\ \cup_{i=1}^N A_i = \mathcal{L} \end{cases}$$

where the objective is the minimization of the largest patrolling time lag among all

areas A_i, and the constraints represent the physical limitations of the cameras and the requirement that all points in $\mathcal{L}$ are eventually visited, respectively. The previous problem can be re-cast as a linear program (LP) as follows (the proof can be found in Alberton et al. (2012)):

Proposition 3.2.2. ***Alberton et al. (2012)*** *The optimization problem $\mathcal{P}_1$ is equivalent to the following LP problem:*

$$\mathcal{P}_1' : \qquad T^*_{\mathcal{P}_1} = \min_{\boldsymbol{\xi},\alpha} 2\alpha$$
$$s.t. \begin{cases} \frac{r_i-\ell_i}{\bar{v}_i} \leq \alpha & i = 1,\dots,N \\ \underline{d}_i \leq \ell_i \leq \overline{d}_i, \quad \underline{d}_i \leq r_i \leq \overline{d}_i & i = 1,\dots,N \\ r_i \geq \ell_{i+1} & i = 1,\dots,N \\ \underline{d}_1 = \ell_1 = 0, \quad \overline{d}_N = r_N = L & \end{cases}$$

Analysing the previous problem, for the cost functional

$$J_\infty(\xi) = \max_i \frac{2(r_i - \ell_i)}{\bar{v}_i}, \tag{3.4}$$

it holds that the minimum value J^*_∞ achievable for $J_\infty(\boldsymbol{\xi})$, with $\boldsymbol{\xi}$ respecting the physical and interlacing constraints, is equal to the optimal solution $T^*_{\mathcal{P}_1}$ of problem $\mathcal{P}_1'$. $\Xi^*_{\mathcal{P}_1}$ denotes the set of minimizers of $J_\infty(\boldsymbol{\xi})$ or, equivalently, of $\mathcal{P}_1'$.

The previous proposition provides a centralized solution to the patrolling problem, but cannot be easily computed in a distributed fashion. Although distributed algorithms exist for the solution of LP problems Notarstefano and Bullo (2011), these involve the solution of the entire problem at each node, which is a futile computational effort. Moreover, the previous optimization problem might have multiple minimizers. Again, it is possible to formulate a new optimization problem, $\mathcal{P}_2$, whose minimizer is unique and is also a minimizer for the original problem $\mathcal{P}_1$. Introducing the following cost functional

$$J_2(\boldsymbol{\xi}) = \sum_{i=1}^{N} \frac{1}{\bar{v}_i}(r_i - \ell_i)^2, \tag{3.5}$$

the following proposition holds (its proof can again be found in Alberton et al. (2012))

Proposition 3.2.3. ***Alberton et al. (2012)*** *Consider the optimization problem*

$$\mathcal{P}_2: \qquad J_2^* = \min_{\boldsymbol{\xi}\in\mathbb{R}^{2N}} J_2(\boldsymbol{\xi})$$
$$s.t. \begin{cases} \underline{d}_i \le \ell_i \le \overline{d}_i, \quad \underline{d}_i \le r_i \le \overline{d}_i & i = 1,\dots,N \\ r_i \ge \ell_{i+1} & i = 1,\dots,N-1 \\ \underline{d}_1 = \ell_1 = 0, \quad \overline{d}_N = r_N = L \end{cases}$$

The corresponding set of minimizers $\Xi^*_{\mathcal{P}_2}$ *is a singleton and* $\Xi^*_{\mathcal{P}_2} \subseteq \Xi^*_{\mathcal{P}_1}$.

The benefits of the optimization problem $\mathcal{P}_2$ as compared to the optimization problem $\mathcal{P}_1$ are mainly two, namely: (i) using specific communication strategies $\mathcal{P}_2$ can be solved with distributed, scalable and parallelizable algorithms; (ii) the uniqueness of the minimizer in $\mathcal{P}_2$ guarantees the practical convergence of iterative numerical algorithms.

Remark 3.2.4. Note that, intuitively speaking, the solution of problem $\mathcal{P}_2$ shares the patrolling burden as evenly as possible among all the cameras. The unique partition that solves $\mathcal{P}_2$ is such that each camera has a time lag that is as similar as possible to the time lag of its neighbors. In some way, it is similar to what happens with the problem of finding x such that $Ax = b$, when $A \in \mathbb{R}^{n\times n}$ is singular and $b \in \mathbb{R}_n$ is a given vector. The problem has many solutions, but the one obtained by using the pseudo inverse of A is the one that minimizes the norm of vector x.

Remark 3.2.5. As pointed out in Pasqualetti et al. (2014), having the cameras sweep the assigned portions of the perimeter at the maximum speed is efficient for static intruders, while for smart dynamic intruders a more sophisticated law is needed. In particular, this law entails the synchronization of the neighboring cameras in such a way that they simultaneously visit the extreme in common. However, this control law can be applied to any partitioning of the environment to be patrolled. Therefore, to better manage smart intruders, one can apply the equal-waiting trajectory algorithm suggested in Pasqualetti et al. (2014) on the partitioning of $\mathcal{L}$ given by the optimal solution of problem $\mathcal{P}_2$. In this way it is possible to combine good performance for both static and dynamic intruders.

Problems $\mathcal{P}'_1$ and $\mathcal{P}_2$ can be rapidly solved using a centralized algorithm. However, a distributed approach has its own advantages, already highlighted in the introduction. Perhaps the more interesting is the capability to adapt to dynamic changes like intruders tracking or the presence of faulty cameras Pasqualetti et al. (2014). These events are usually only local, and in a centralized approach, each time a new intruder appears or each time a camera fails the algorithm has to be reset for the whole network.

Now that the problem have been presented, the set-up for the optimization is described. The communication graph among the agents (that is the cameras) is very particular. In

fact, according to the cameras' disposition along the perimeter, each camera communicates only with the previous and the following one, that is camera $i = \{2, \ldots, N-1\}$ exchanges information with cameras $i-1$ and $i+1$, while camera 1 (camera N) communicates only with camera 2 (camera $N-1$ respectively). The corresponding communication graph is therefore a line graph.

The algorithm designed to solve Problem $\mathcal{P}_1$ has to have the following features:

1. *Asymptotic local estimation*: each camera i has to asymptotically estimate only its own optimal patrolling area, defined by ℓ_i^* and r_i^*.
2. *Peer-to-peer (leaderless)*: each cameras' update has to consider the limited computational and memory capability available at the camera itself and there is no "master" camera. Moreover, the algorithm can only require communication between one-hop neighbors.
3. *Distributed*: the update-rule of the local variables at each camera has to depend only on the variables stored by the camera and by its neighbors. No multi-hop information exchange is allowed.
4. *Asynchronous*: the algorithm has to allow the cameras to perform the update step and the communication step in any moment, without any coordination among the agents.
5. *Lossy broadcast communication without ACK*: the convergence of the algorithm has to be assured even if communication is lossy and broadcast-based. No ACK mechanisms has to be employed.

3.3 A coordinated broadcast partitioning algorithm (CB algorithm)

The aim of this section is to introduce a distributed algorithm to solve problem $\mathcal{P}_2$ which works with a reliable communication scenario. Concerning the communication protocol, the one employed is a combination of an asymmetric broadcast and a coordinated broadcast. In fact, at each time step there is one camera that wakes up, sends some information to its neighbors (and so it is an asymmetric broadcast), but its neighbors are also supposed to send the result of their computation back to the node.

The algorithm proposed is not developed according to standard optimization algo-

rithms. In a way it is an intuitive procedure to solve the problem that can be shown to be effective. In order to solve the problem, the update of the patrolling area has to be such that (i) the physical constraints and the covering constraint are satisfied at each iteration, and (ii) the set of patrolling areas converges to the optimal partition.

The strategy proposed is next described and reported as Algorithm 3.1. Suppose the patrolling areas are initialized in such a way that the physical and interlacing constraints are satisfied and let the iterations of the algorithm be indexed by the discrete time variable $k \in \mathbb{N}$. In the following the algorithm is described in case the activated camera is neither the first one nor the last one. However, if the selected camera is $i = 1$ ($i = N$) an *ad hoc* adjustment has to be done, i.e. only the update of camera $i+1$ (resp. $i-1$) has to be done. Assume that at iteration k camera i is activated and transmits the values of $\ell_i(k)$ and $r_i(k)$ to its neighboring cameras $i-1$, $i+1$. Based on the information received, cameras $i-1$ and $i+1$ update the extremes of their patrolling areas that are "closer" to camera i, namely, r_{i-1} and ℓ_{i+1}, respectively. For simplicity, only the update performed by camera $i-1$ is described.

Let $m_{i-1}(k)$ and $m_i(k)$ be the middle points of $A_{i-1}(k)$ and $A_i(k)$, respectively, i.e,

$$m_{i-1}(k) = \frac{\ell_{i-1}(k) + r_{i-1}(k)}{2}, \quad m_i(k) = \frac{\ell_i(k) + r_i(k)}{2}. \tag{3.6}$$

Camera $i-1$ computes the point c_ℓ^* which splits the segment $[m_{i-1}(k), m_i(k)]$ into two parts that require the same time to be swept by the respective cameras. Mathematically,

$$c_\ell^* = \frac{\bar{v}_i(\ell_{i-1}(k) + r_{i-1}(k)) + \bar{v}_{i-1}(r_i(k) + \ell_i(k))}{2(\bar{v}_i + \bar{v}_{i-1})}$$

Camera $i-1$ then sets $r_{i-1}(k+1) = c_\ell^*$, provided that this update does not violate the physical constraints, i.e, it must holds $c_\ell^* \in \left[\underline{d}_i, \overline{d}_{i-1}\right]$; otherwise $r_{i-1}(k+1)$ is set equal to the closest point to c_ℓ^* that satisfies the physical constraint (see lines 4 through 10). Finally camera $i-1$ sends the value $r_{i-1}(k+1)$ to camera i which updates its left extreme accordingly, that is, $\ell_i(k+1) = r_{i-1}(k+1)$ (see line 22). Camera $i+1$ carries out an analogous update: in this case $\ell_{i+1}(k+1)$ and $r_i(k+1)$ are the extremes involved (see lines 12 through 19 and line 23).

Figure 3.2 shows one step of the execution of the algorithm. Observe that each iteration of the CB algorithm involves two communication rounds; the first one from camera i to cameras $i-1$ and $i+1$, referred to as the *forward communication*, and the second one from cameras $i-1$ and $i+1$ to camera i, referred to as *backward communication*.

The convergence properties of the CB algorithm can be characterized by the following

Algorithm 3.1 CB algorithm (time k, camera i activated)

Broadcast forward communication: camera i transmits $r_i(k)$ and $\ell_i(k)$ to cameras $i+1$ and $i-1$.
{% Update of the right extreme of camera $i-1$}
$c^*_\ell = \frac{\bar{v}_i(\ell_{i-1}(k)+r_{i-1}(k))+\bar{v}_{i-1}(r_i(k)+\ell_i(k))}{2(\bar{v}_i+\bar{v}_{i-1})}$;
if $c^*_\ell < \underline{d}_i$ **then**
  $r_{i-1}(k+1) = \underline{d}_i$;
else if $c^*_\ell > \overline{d}_{i-1}$ **then**
  $r_{i-1}(k+1) = \overline{d}_{i-1}$;
else
  $r_{i-1}(k+1) = c^*_\ell$;
end if
{% Update of the left extreme of camera $i+1$}
$c^*_r = \frac{\bar{v}_{i+1}(\ell_i(k)+r_i(k))+\bar{v}_i(\ell_{i+1}(k)+r_{i+1}(k))}{\bar{v}_i+\bar{v}_{i+1}}$;
if $c_r* > \overline{d}_i$ **then**
  $\ell_{i+1}(k+1) = \overline{d}_i$;
else if $c^*_r < \underline{d}_{i+1}$ **then**
  $\ell_{i+1}(k+1) = \underline{d}_{i+1}$;
else
  $\ell_{i+1}(k+1) = c^*_r$;
end if
{% Update of the extremes of camera i}
Peer to peer backward communication: camera i receives
  from its neighbours $\ell_{i+1}(k+1)$ and $r_{i-1}(k+1)$.
$\ell_i(k+1) = r_{i-1}(k+1)$;
$r_i(k+1) = \ell_{i+1}(k+1)$;

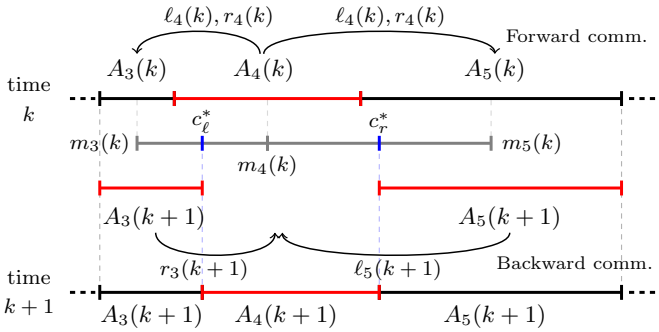


Figure 3.2: Execution of one step of the algorithm in a simplified set-up with $D_i = \mathcal{L}$ and equal $\bar{v}_i$ for all i. The camera activated at time k is camera 4.

result.

Theorem 3.3.1. *Let $\boldsymbol{\xi}(0)$ describe the initial patrolling areas, satisfying the physical*

and interlacing constraints. Assume Assumption 2.3.2 holds true. Then the trajectory $\{\boldsymbol{\xi}(k)\}_k$ *generated by the CB algorithm satisfies that*

1. *the physical, interlacing and covering constraints are verified for all* $k \in \mathbb{N}$*;*
2. *the cost functional* J_2 *is non increasing and satisfies*

$$J_2(\boldsymbol{\xi}(k+1)) < J_2(\boldsymbol{\xi}(k)), \qquad \text{if } \boldsymbol{\xi}(k+1) \neq \boldsymbol{\xi}(k).$$

3. *the cost functional* J_∞ *is non increasing and satisfies*

$$J_\infty(\boldsymbol{\xi}(k+\bar{\tau})) < J_\infty(\boldsymbol{\xi}(k)), \qquad \text{if } \boldsymbol{\xi}(k) \notin \Xi^*_{\mathcal{P}_1},$$

   where $\bar{\tau} = (N-1)(\tau+1)$*;*

4. *the cost functionals* J_2 *and* J_∞ *converges, respectively, to* J_2^* *and* J_∞^**.*

A detailed proof can be found in Appendix B.1. Looking at the iterations of the algorithm as the evolution of a dynamical system, the proof aims at showing that this system meets the assumptions of Theorem 4.3 in Bullo, Carli, and Frasca (2012), which regards convergence of set-valued dynamical systems. The proof mainly reduces to show that J_2 is a Lyapunov function for the system. Having J_2 Lyapunov function has the advantage that the solution eventually reached is unique (since J_2 has a unique minimizer). In fact, as a consequence of the fourth item in the previous theorem, it is possible to state the following corollary:

Corollary 3.3.2. *Under the same hypotheses of Theorem 3.3.1, the trajectory* $\{\boldsymbol{\xi}(k)\}_k$ *generated by the CB algorithm converges to the optimal solution of Problem* $\mathcal{P}_2$*, i.e,* $\boldsymbol{\xi}(k) \to \boldsymbol{\xi}_2^*$ *as* k *tends to infinity, and, in turn, to an optimal solution of* $\mathcal{P}_1$*.*

3.4 r-CB: a version robust to packet losses

CB algorithm assumes that the communication channels are reliable and, in particular, that no packet losses occur. In this section this assumption is relaxed and so transmission failures are allowed in the communication between neighboring cameras. In presence of unreliable communications, the CB algorithm presents a major shortcoming, as explained in the following. Observe that, during each iteration of the CB algorithm, there are two possible "sources" of packet loss: (i) the packet broadcast by camera i during the forward communication is not received by camera $i-1$ (or analogously by camera $i+1$);

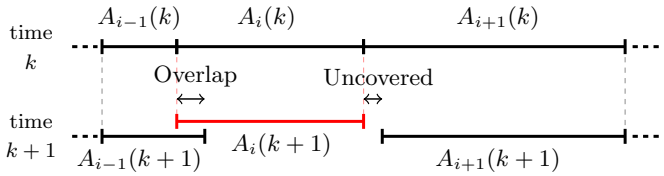


Figure 3.3: Consequences of the failure of the backward communication: generation of an overlap between the patrolling areas and of an uncovered part of the environment. The situation at time k corresponds to that presented in Figure 3.2

in this case, the respective extremes remain unchanged and nothing happens; (ii) the packet sent by camera $i-1$ (or analogously by $i+1$) to camera i during the backward communication is not received, and, in turn, camera i does not update the respective extreme; it might result that $r_{i-1}(k+1) \neq \ell_i(k+1)$ and the interlacing and covering constraints might be violated (see Figure 3.3).

The latter failure is the most critical one; indeed it might cause the presence of parts of the perimeter that are left unassigned and so are uncovered. To deal with such presence of uncovered areas, it is possible to modify the CB algorithm. Specifically, consider iteration k and assume that the interlacing constraints (3.2) among all the cameras are satisfied. Moreover assume that camera i is the camera performing the forward communication round. If camera $i-1$ receives the information related to $\ell_i(k)$ and $r_i(k)$, then it computes c^*_ℓ as done for the CB algorithm, and it updates r_{i-1} as follows

$$r_{i-1}(k+1) = \begin{cases} \ell_i(k) & \text{if } c^*_\ell \leq \ell_i(k) \\ \min\left\{c^*_\ell, \overline{d}_{i-1}\right\} & \text{if } c^*_\ell > \ell_i(k) \end{cases} \tag{3.7}$$

Then camera $i-1$ sends the value $r_{i-1}(k+1)$ to camera i; if the packet is received, then camera i sets $\ell_i(k+1) = r_{i-1}(k+1)$, otherwise ℓ_i remains unchanged, i.e., $\ell_i(k+1) = \ell_i(k)$. Observe that, according to the update proposed in (3.7), it holds that $\ell_i(k+1) \leq r_{i-1}(k+1)$, and, hence, the interlacing constraint between cameras $i-1$ and i is still satisfied. This new algorithm robust to packet losses, is denoted hereafter as r-CB (its algorithmic description is given in Algorithm 3.2).

To characterize the convergence properties of r-CB an assumption on the frequencies of transmission failures is needed. It is enough to use Assumption 2.3.1 with a slight modification, in particular there has to be a limit on the number of consecutive communication failures between camera i and j only regarding the forward communication, while for the backward communication the packet losses can be unbounded. Clearly, this is not really an advantage with respect to considering all the communication types to have a bounded consecutive packet loss but it is interesting to notice nevertheless.

Algorithm 3.2 r-CB algorithm (time k, camera i activated)

Broadcast forward communication: camera i transmits $r_i(k)$ and $\ell_i(k)$ to cameras $i+1$ and $i-1$.

Update if camera $i-1$ receives information

$c_\ell^* = \frac{\bar{v}_i(\ell_{i-1}(k)+r_{i-1}(k))+\bar{v}_{i-1}(r_i(k)+\ell_i(k))}{2(\bar{v}_i+\bar{v}_{i-1})}$;
if $c_\ell^* < \ell_i(k)$ **then**
  $r_{i-1}(k+1) = \ell_i(k)$;
else
  $r_{i-1}(k+1) = \min\left\{c_\ell^*, \bar{d}_{i-1}\right\}$;
end if

Update if camera $i+1$ receives information

$c_r^* = \frac{\bar{v}_{i+1}(\ell_i(k)+r_i(k))+\bar{v}_i(\ell_{i+1}(k)+r_{i+1}(k))}{\bar{v}_i+\bar{v}_{i+1}}$;
if $c_r* > r_i(k)$ **then**
  $\ell_{i+1}(k+1) = r_i(k)$;
else
  $\ell_{i+1}(k+1) = \max\{c_r^*, r_i(k)\}$;
end if

The algorithm performs steps $20 \div 23$ of Algorithm 3.1, provided the backward communications are successful.

The convergence results of r-CB are given in this theorem:

Theorem 3.4.1. *Let $\boldsymbol{\xi}(0)$ describe the initial patrolling areas, satisfying the physical and interlacing constraints, and let Assumptions 2.3.2 and 2.3.1 hold true. Then, the trajectory $\{\boldsymbol{\xi}(k)\}_k$ generated by the r-CB algorithm satisfies that*

1. *the physical, interlacing and covering constraints are verified for all $k \in \mathbb{N}$;*
2. *the cost functional $J_\infty(k)$ is not increasing and satisfies*

$$J_\infty(\boldsymbol{\xi}(k+\tau_{\max})) < J_\infty(\boldsymbol{\xi}(k)) \qquad \text{if } \boldsymbol{\xi}(k) \notin \Xi_\infty^*.$$

   where $\tau_{\max} := 2h\tau(N-1)+1$.
3. *$J_\infty(\boldsymbol{\xi}(k))$ converges to J_∞^*.*

The detailed proof is reported in Appendix B.2. In this case, the situation is a bit more complicated than the one for the CB algorithm. The Lyapunov function employed in the proof of the CB algorithm is not Lyapunov anymore for the r-CB one. It is therefore necessary to study the evolution in time of J_∞, which, conversely to J_2, does not have a unique minimizer. The proof relies on the introduction of a theorem similar to that in Bullo et al. (2012), and on showing that, if the current partitioning is not

optimal, J_∞ is strictly decreasing after a bounded number of consecutive iterations of the r-CB algorithm. Since J_∞ does not have a unique minimizer, in this case it is possible to show only convergence to the set of minimizers of problem $\mathcal{P}_1$:

Corollary 3.4.2. *Under the same assumptions of Theorem 3.4.1, the trajectory* $\{\boldsymbol{\xi}(k)\}_k$ *generated by the r-CB algorithm converges to the set of optimal solutions of problem* $\mathcal{P}_1$*, i.e.,* $\boldsymbol{\xi}(k) \to \Xi^*_\infty$ *as* k *tends to infinity.*

Remark 3.4.3. The algorithm presented in this work is similar to the one presented in Alberton et al. (2012); Borra et al. (2015). However, the mathematical machinery used here is substantially different from the one employed in Alberton et al. (2012); Borra et al. (2015), which considers only the lossless scenario and which strongly relies on the monotonicity of $J_2(\boldsymbol{\xi})$ and on its minimum being unique. In fact, when packet loss is considered neither $J_2(\boldsymbol{\xi})$ nor $J_\infty(\boldsymbol{\xi})$ satisfy the hypotheses of the theorems in Borra et al. (2015). Moreover, Theorem 3.4.1 is rather general and might be applicable to other relevant applications such as 2D/3D partitioning in cooperative robotics with asynchronous and lossy communication.

Some simulations are also presented to further show the effectiveness of the algorithm. The setting for the first simulation is the following: the number N of cameras takes different values, the length of the environment is $L = 10N$ and the maximum speed is $\bar{v}_i = 2$ for all cameras. The patrolling range of cameras $i = 2, \dots, N-1$ is $[10(i-1)-2, 10i+2]$, for camera 1 is $[0, 12]$ and for camera N is $[10(N-1)-2, 10N]$. Concerning the communication reliability, a communication works with a probability of 70% and the value for threshold h is 10 (in the implementation it is assured that Assumption 2.3.1 is satisfied); also, every N iterations all the cameras are activated once, implying a value for parameter τ in Assumption 2.3.2 equal to N. The initialization for the algorithm is $l_i(0) = \underline{d}_i$, $r_i(0) = \overline{d}_i(0)$ for all the cameras. Figure 3.4 shows the normalized cost functions for a realization of the r-CB algorithm. Given a cost function $J(k)$ with optimal value J^*, its normalized form $\tilde{J}$ is given by

$$\tilde{J}(k) = \frac{J(k) - J^*}{J(0) - J^*}. \tag{3.8}$$

The figure confirms that, as demonstrated in Theorem 3.4.1, $\tilde{J}_\infty$ does not increase as the number of iterations increases and converges to the optimal value, while for $\tilde{J}_2$ the non-increasing property does not hold. This clearly shows that $J_2(k)$ cannot be used as a Lyapunov function. Nevertheless, $J_2(k)$ still seems to converge to its optimal value. It is therefore possible to conjecture that r-CB still converges to a unique point (the unique optimizer of problem $\mathcal{P}_2$), which is not so strange intuitively since the modification only

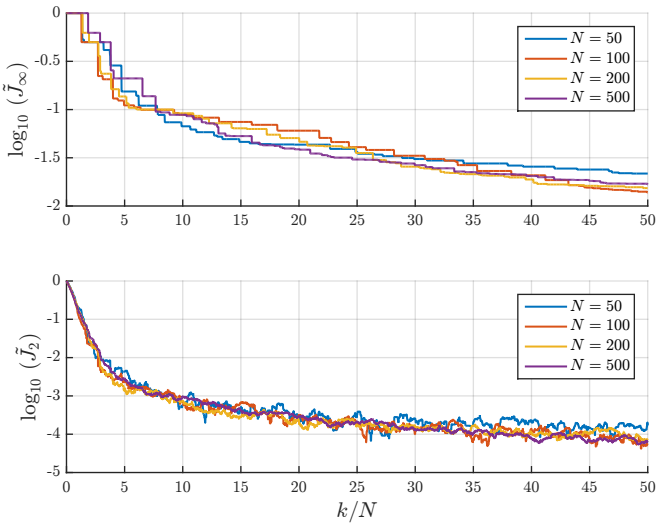


Figure 3.4: Logarithm of the normalized cost functions $\tilde{J}_\infty$ and $\tilde{J}_2$ (see Equation (3.8)) for one realization of the r-CB algorithm and different values of N. The time scale is divided by N, which implies that at each integer value of k/N all the cameras have been activated once.

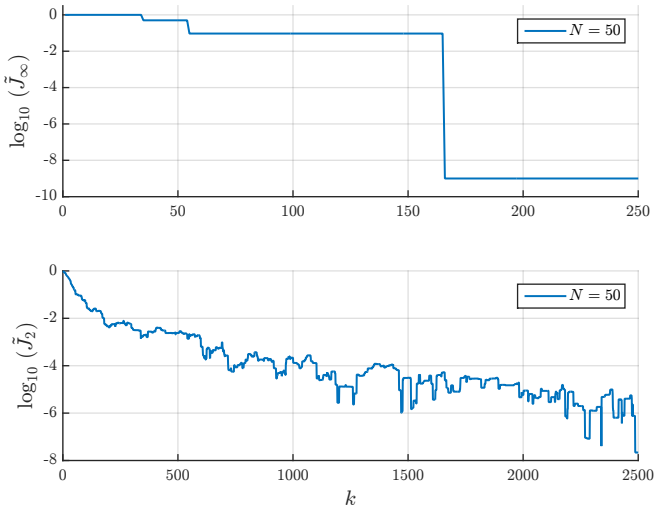


Figure 3.5: Logarithm of the normalized cost functions $\tilde{J}_\infty$ and $\tilde{J}_2$ (see Equation (3.8)) for one realization of the r-CB algorithm for $N = 50$ in the second set-up considered. The time scale is NOT divided by N, but it is different for the two plots. After $k = 166$ in the first plot, J_∞ has achieved its minimum value J^*_∞, and for simplicity this is represented by letting $\log_{10}(\tilde{J}_\infty) = -9$.

slightly varies the update of the CB algorithm. Note that the speed of convergence of the algorithm is similar for all the sizes of the network considered, which implies that the algorithm is very scalable. Together with scalability, the algorithm has a really moderate computational burden and so can be very easily implemented in a multi-agent set -up.

Figure 3.5 contains an additional simulation. In this case, the maximum speed and the physical limits of each camera are randomly chosen (always respecting the interlacing physical coverage constraints (3.1)). In the previous case, due to the symmetry of the set-up, the physical constraints do not play any role in determining the optimal solution, since the extremes of each optimal patrolling area strictly satisfy

$$\underline{d}_i < \ell_i < \overline{d}_{i+1}, \quad \underline{d}_{i+1} < r_i < \overline{d}_i$$

for all cameras. The solution is reached only asymptotically, since it is obtained through a redistribution method. Conversely, in the second case, the differences in speed and physical constraints create a different situation. In particular, in Figure 3.5, what happens is that the value J_∞^* is determined by the sweeping time of camera 44, whose optimal value of the extremes ℓ_{44} and r_{44} of its patrolling areas correspond respectively to $\overline{d}_{43}$ and $\underline{d}_{45}$, the physical constraints of its neighbor. This means that there is no way that the sweeping time of camera 44 can be diminished below that value, since the neighbors cameras cannot further help camera 44. In this case the algorithm reaches in a finite and very short time the minimum value for J_∞ (every camera is activated less than 4 times in order to reach its minimum), which is determined by camera 44. On the other hand, the value of J_2 continues to decrease (as a trend), because the remaining cameras continue to divide as equally as possible their patrolling burden.

3.5 Final considerations on the patrolling problem

This chapter focused on an application for smart camera networks, a multi-agent system which is nowadays highly utilized especially for security reasons. The advantages of the use of a distributed approach for this specific application have been highlighted, and can be summarized into good problem scalability, safety and possibility to easily adapt to dynamic changes. The convergence is shown for both algorithms introduced, but the differences in the two proofs shows that for the robust version of the algorithm the demonstration is more complicated and it is not possible to show convergence to a unique point, but only to a set.

4

Minimization of locally coupled cost functions

The results of this chapter are the subject of the following submitted paper
Todescato M., Bof N., Cavraro G., Carli R., and Schenato L. Generalized gradient optimization over lossy networks for partition-based estimation. *arXiv preprint, arXiv:1710.10829*

The algorithm developed in this chapter can be used to solve a particular class of problems that arises in multi-agent systems, when each agent is endowed with a private cost. It has therefore a wider applicability than the algorithms introduced in Chapter 3, where the problem was really specific. In the class of problems here analyzed, each agent has its own optimization variable, but its cost function depends not only on its own optimization variable but also on those of its one-hop neighbors. The agents' aim is to minimize the sum of the private cost functions, using the communication set-up defined in Section 2.2, an asynchronous protocol and not assuming perfect communication. Even though many algorithms exist to solve similar problems, at the best of the author's knowledge there is none which address simultaneously all the features that are desired. The algorithm developed is applied to solve an estimation task for smart power grids, but the same algorithm can be useful also in other multi-agent systems like robot networks, sensor networks and so on (as long as the function to minimize is locally coupled).

4.1 Introduction and state of the art

The problem considered in this chapter has evident similarities with the one analyzed in Chapter 6. In both cases, given a set of agents, each is endowed with a cost function, and the agents' aim is to minimize the sum of these cost functions. The difference between the two chapters is in the specific cost functions utilized.

In particular, as already described in the introduction, on the one hand this chapter deals with the minimization of locally coupled cost functions (see Figure 1.19), and, on the other hand, Chapter 6 deals with additively separable cost functions, where the agents have to reach consensus on the optimization variable (see Figure 1.23). The state of the art presented below, briefly involves also some works that concern the minimization of additively separable cost functions, and that will be useful in Chapter 6.

It is possible to find many papers that tackle the distributed minimization of the sum of cost functions. This optimization has to be carried out by a group of agents, each of them contributing to the final cost function with its own private term. There are mainly three kinds of distributed algorithms to solve such problems.

The first class of algorithms relies on primal sub-gradient or descent iterations, as in Nedić and Ozdaglar (2009); Nedić, Ozdaglar, and Parrilo (2010); Marelli and Fu (2015). These methods have the advantage to be easy to implement and suitable for asynchronous computation. In the first two cited works, the problem is the minimization of additively separable cost functions, with all the agents that have to agree on the choice of the minimizer. The third work, Marelli and Fu (2015), concerns instead a problem very similar to the one studied in this chapter.

A second class of algorithms involves dual variables. In particular, augmented Lagrangian algorithms such as the Alternating Direction Methods of Multipliers (ADMM) recently became popular, especially because they usually present a good convergence speed. ADMM was developed as a centralized algorithm, but can be used to manage a distributed set-up (in Chapter 5 the idea behind its distributed version will be explained). Boyd et al. (2011) contains a survey on ADMM and its wide applicability. However, most of the ADMM distributed algorithms are based on a consensus iteration Wei and Ozdaglar (2013). Thus, each node must store in its local memory a copy of the entire state vector. To avoid this problem, a recent partition-based and scalable approach applied to the ADMM algorithm is presented in Erseghe (2012). A similar idea is presented in Kekatos and Giannakis (2013). Most of the previous works requires a synchronous communication set-up, and only recently suitable modification of the ADMM algorithm have been proposed to cope with an asynchronous set-up Wei and Ozdaglar (2013); Iutzeler, Bianchi, Ciblat, and Hachem (2013); Bianchi, Hachem, and Iutzeler (2014). In

all the latter works, the agents have to agree to the same minimizer.

There exist also other distributed algorithms for the solution of the problem (in case consensus on the minimizer is sought), which are based on Newton methods Zargham, Ribeiro, Ozdaglar, and Jadbabaie (2014); Zanella, Varagnolo, Cenedese, Pillonetto, and Schenato (2011). The algorithm presented in the latter work will be the starting point for Chapter 6.

This chapter addresses the minimization of a global cost function which is the **sum of convex locally coupled local costs** (see Figure 1.19). The minimization (which is unconstrained) has to be carried out in a distributed way by a group of agents (each owner of one of the local costs). The algorithm developed has to cope with an **asynchronous** and possibly **lossy** communication set-up. The aim is to actually exploit the fact that the costs are locally coupled and to have each agent evaluate only the part of the optimization variable which it owns (so consensus among agents is not sought). Moreover, each agent's update has to exploit information coming only from one-hop neighbors.

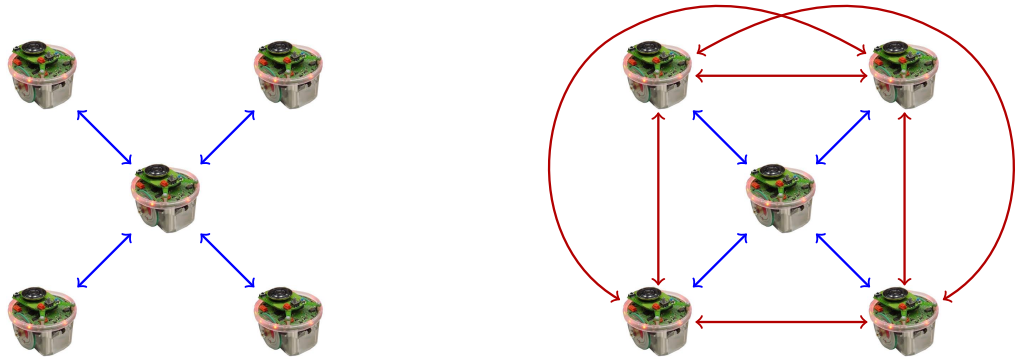
The problem itself is interesting since its structure characterizes a large variety of applications such as multi-area electric grid state estimation Conejo, de la Torre, and Canas (2007); Bolognani, Carli, and Todescato (2014), localization in multi-robots formation and sensor networks Carron, Todescato, Carli, and Schenato (2014); Bof, Todescato, Carli, and Schenato (2016a) and Network Utility Maximization Palomar and Chiang (2006). As a consequence, the algorithm developed to solve this problem can be applied in different situations.

From an algorithmic point of view, the solution proposed, which is gradient descent based, has some peculiarity with respect to other gradient-based algorithms like Nedić and Ozdaglar (2009); Nedić et al. (2010); Marelli and Fu (2015). In particular, for the class of cost functions considered in this chapter, the solution given by the works just cited (and by similar approaches) usually does not lead to a distributed algorithm (as intended in this thesis). Indeed, even though the cost is the sum of locally coupled costs, its derivatives are not usually locally coupled (meaning that they only depend on the neighbors' information), but depend on information related to multi-hop processing units. Hence, the local functional dependence cannot be directly exploited. To overcome this issue, in some cases (Nedić and Ozdaglar (2009); Nedić et al. (2010)) the algorithms require the local exchange of global information, hence all the agents eventually reach consensus to an optimal solution. Conversely, the algorithm presented here allows each agent to work only on its own part of the solution. In other cases, the algorithms require multiple communication rounds within the same algorithmic iteration (Marelli

and Fu (2015)), with this latter solution implicitly asking for synchronicity. Instead, the algorithm introduced here requires just one communication exchange per iteration, and this exchange is allowed only between one-hop neighbors.

Another very important aspect is that the solution proposed works in a lossy communication scenario. To do so it employs what might look like a natural approach: the processing units store the last successfully received information from the neighboring nodes. This idea to solve the problem of packet losses is similar to the one adopted in the *partially asynchronous iterative methods* Bertsekas and Tsitsiklis (1989); Tsitsiklis, Bertsekas, and Athans (1986). However, as later described in Section 4.3, in the algorithm developed here, because of packet drops, the same state variables appears in multiple delayed version in the same update, which is not allowed in partially asynchronous iterative methods (see Equation 1.2 in Bertsekas and Tsitsiklis (1989)[Chp 7]). Thus, it is not possible to reduce the algorithm proposed in this chapter to these latter methods. Moreover, regarding the problem of computing non-locally coupled derivatives, in partially asynchronous methods each computational unit uses information also from multi-hop neighbors, that is in the update the variables used belong not only to neighbors, but only to the neighbors of the neighbors. As so, information is not shared only between the neighbors in the communication graph, but also among neighbors in a hyper-communication graph (using for example multi-hop communication). For example, consider the case where the communication graph has a star topology. According to this graph, each peripheral node can communicate only with the central node, while the central processor can communicate with everyone else, and so each peripheral node has information coming only from the central node, and the central node has information from everyone else. Conversely, if a two-hop communication is exploited, then the nodes use information coming from all the others, and the hyper-communication graph is *complete*. Figure 4.1 shows the two situations. This latter aspect was perhaps less relevant in the context explored in Bertsekas and Tsitsiklis (1989), especially because it focused on the parallelization of the computation. In a peer-to-peer situation (that is in a multi-agent system) this aspect is much more important.

The main contribution of the paper is a truly distributed algorithm, based on a modified *generalized gradient descent* iteration which, under suitable assumptions on the step size, is provably convergent and which is resilient to the presence of packet losses in the communication channel. To the best of the author's knowledge, this is one of the first provably convergent algorithms in the presence of packet losses, since even if both ADMM algorithms and distributed sub-gradient methods (DSM) can handle asynchronous computations, they still require reliable communication and usually require



Star graph

Hyper-graph considering two-hops neighbors

Figure 4.1: On the left the starting communication graph which has a star topology. On the right, the hyper-graph obtained if two-hops neighbors are allowed to exchange information. The red arrows are those added from the starting graph.

communication among two-hops neighbors (as will be later on showed). Some preliminary versions of this algorithm appeared in Todescato, Cavraro, Carli, and Schenato (2015) for the specific case of quadratic programming, and in Bof et al. (2016a) for the specific application of sensor networks locations.

The proposed algorithm is suitable for fully parallel computation, i.e., multiple agents can communicate and update their local variable simultaneously, and broadcast communication can be employed, i.e., nodes do not need to enforce a bidirectional communication such as in gossip algorithms. It is therefore very attractive from a practical point of view.

The algorithm is inspired on and resembles the block Jacobi iteration appeared in Bertsekas and Tsitsiklis (1989); Bin and Lin (1994). However, in Bertsekas and Tsitsiklis (1989), the authors are majorly interested in parallel computation rather than implementing a distributed procedure suitable for today's sensor networks, ans so an hyper-communication graph is exploited. Conversely, in Bin and Lin (1994) the particular local dependency considered in the cost function ensures that first and successive derivatives are locally coupled. The proposed procedure is quite flexible, and from the latter it is possible to derive a gradient descent iteration and a resilient generalized gradient for quadratic programming. In this second case, the algorithm has global conergence.

The algorithm is finally tested on the standard IEEE 123-nodes test feeder for robust state estimation in presence of measurements outliers.

4.2 Problem formulation

Consider a set of N agents $\mathcal{V} = \{1, \dots, N\}$, where each agent $i \in \mathcal{V}$ is described by its state vector $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ and assume that they can communicate accordingly to communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which has to meet the following assumption

Assumption 4.2.1. The communication graph $\mathcal{G}$ is time-invariant, **undirected** and satisfy the Assumption 2.2.1 on connectivity.

Defining the overall state vector as $\boldsymbol{x} := [\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_N^\top]^\top \in \mathbb{R}^n$ ($n = \sum_i n_i$), the optimization problem to be solved is

$$\min_{\boldsymbol{x}} J(\boldsymbol{x}) \equiv \min_{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N} \sum_{i=1}^{N} J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j \in \mathcal{N}_i}) \,. \tag{4.1}$$

Observe that the J_is' local dependence coincides with the communication graph $\mathcal{G}$, i.e., each cost function J_i depends on information regarding only agent $j \in \mathcal{N}_i^+$.

The convergence of the algorithm proposed will be assured it the following assumption on the total cost function is met:

Assumption 4.2.2 (Strict convexity and radial unboundedness)**.** The function $J(\boldsymbol{x})$ is assumed to be strictly convex and radially unbounded.

Under the previous assumption the minimizer $\boldsymbol{x}^*$ of Problem (4.1) exists and is unique

$$\boldsymbol{x}^* := \operatorname*{argmin}_{\boldsymbol{x}} \; J(\boldsymbol{x}) \,, \tag{4.2}$$

but the local costs function J_is do not need to be strictly convex and radially unbounded. Indeed in many estimation problems the local cost functions J_is are just strictly convex but not radially unbounded. The standard approach to solve the previous optimization problem is to resort to some centralized iterative algorithm acting on J, e.g., Newton-Raphson, which makes use of global knowledge of the network' states, costs and topology.

Conversely, the algorithm developed in this chapter has to respect some features, which limit the agents' possibility to obtain global knowledge on the network. These features are described in the following.

The algorithm designed to solve Problem (4.1) has to have the following features:

1. *Asymptotic local estimation*: each agent i has to asymptotically estimate only its part $\boldsymbol{x}_i^*$ of the overall optimal solution $\boldsymbol{x}^*$.
2. *Peer-to-peer (leaderless)*: each node's update has to consider the limited computational and memory capability available at the node itself and there is no master node among the agents. Moreover, the algorithm can only require communication between one-hop neighbors and it has to assure convergence on any communication graph $\mathcal{G}$ satisfying Assumption 4.2.1.
3. *Distributed*: the update-rule of the local variables at each node has to depend only on the variables stored by the local node and by its neighbors. No multi-hop information exchange is allowed.
4. *Asynchronous*: the algorithm has to allow the agents to perform the update step and the communication step in any moment, without any coordination among the agents.
5. *Lossy broadcast communication without ACK*: the convergence of the algorithm has to be assured even if communication is lossy and broadcast-based. No ACK mechanisms has to be employed.

To simplify the notation, the local components of gradients and Hessians will be denoted as:

$$\nabla_i J_j = \frac{\partial J_j}{\partial \boldsymbol{x}_i}, \qquad \nabla^2_{i\ell} J_j = \frac{\partial^2 J_j}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_\ell}.$$

Remark 4.2.3. The class of functions considered can arise in diverse applications such as state estimation in smart electric grids Todescato et al. (2015) and sensor networks localization Bof et al. (2016a). In these applications usually a quadratic cost on the residuals is applied, leading to a standard linear least-squares framework. Nevertheless, as shown in Section 4.6, the class of functions that can be used is much more general and comprises penalty functions used, e.g., to perform robust statistics and general nonlinear least-squares optimization. Also, this set-up may arise in parallel computation, if, especially for privacy but also for efficiency reasons, each agent is given only one J_i. In this case, to preserve privacy, thanks to local exchange of information, the machines must distributely compute a solution of (4.1).

4.3 Motivating example: state estimation in smart power distribution grids

Before proceeding to introduce the algorithm, the application on which it is tested at the end of the chapter is described. The tackled problem is state estimation in smart power distribution grids. For the ease of exposition, voltages and currents at a node in the grid are expressed as a real number even though they are phasors, i.e., should be represented as complex numbers. The discussion can be extended w.l.o.g. also to the more realistic scenario (which is indeed considered in Section 4.6 below).

In steady state the voltages and currents in a power distribution grid with N_b busses are regulated by the Kirchhoff's laws which can be written as follow:

$$L\boldsymbol{v} = \boldsymbol{i}^c \,.$$

$L \in \mathbb{R}^{N_b \times N_b}$ is the admittance matrix, and $\boldsymbol{v} \in \mathbb{R}^{N_b}$ and $\boldsymbol{i}^c \in \mathbb{R}^{N_b}$ are the vectors collecting all the N_b voltages and currents of the busses in the grid, respectively. The admittance matrix is a sparse matrix, in the sense that the current at a specific bus ℓ, namely i^c_ℓ, depends only on its own voltage and the voltages of its physically connected neighbor busses in $\mathcal{N}_\ell$, i.e.

$$i^c_\ell = \sum_{j \in \mathcal{N}^+_\ell} L_{\ell j} v_j \,.$$

In future smart distribution grids, it is expected that each bus ℓ will be able to take noisy measurements of its voltage and current, i.e.

$$\begin{aligned} y^v_\ell &= v_\ell + w^v_\ell \,, \\ y^{i^c}_\ell &= i^c_\ell + w^{i^c}_\ell = \sum_{j \in \mathcal{N}^+_\ell} L_{\ell j} v_j + w^{i^c}_\ell \,, \end{aligned}$$

where $w^v_\ell, w^{i^c}_\ell$ represent the measurement noise for the voltage and current measurements at bus ℓ, respectively.

The (centralized) state estimation problem assumes that all these measurements are collected at a central unit, which than evaluates the best estimate of all the voltages and currents $\{v_\ell\}_{\ell=1}^{N_b}$. Usually, the unknown quantities to be estimated are the voltages $\boldsymbol{v}^*$, and from these the currents can be estimated via the Kirchhoff's law $\boldsymbol{i}^{c*} = L\boldsymbol{v}^*$.

In this work, the interest is for solving this problem in a distributed fashion via a partition-based communication architecture. If one consider each bus in the power grid as an agent (which has to solve the estimation problem), the situation is as follows: each

agent has its own (current and voltage) measurements and can communicate with its physically connected neighbors. In fact, it is expected that, in a smart grid, the busses will be embedded with communication capabilities, such as power line communication (PLC), which allow them to communicate with their physically connected neighbors. As so, the communication network and the physical network will coincide.

Even though in the following it is assumed that each bus is a node for the distributed algorithm, this is not compulsory. As a matter of fact, one can consider the presence of N computational units, each associated to a group of busses. This computational unit collects the measurements obtained by the busses, performs the voltage estimation and for example decide some control action to apply on its own part of the power grid.

Assume that each bus is a node (or agent), that is $N = N_b$. Agent ℓ is described by its voltage $x_\ell \in \mathbb{R}$, its measurements $\boldsymbol{y}_\ell := [y_\ell^v \; y_\ell^{i^c}]^\top \in \mathbb{R}^2$ and corresponding measurement errors $w_\ell := [w_\ell^v \; w_\ell^{i^c}]^\top \in \mathbb{R}^2$. Define also vectors $\boldsymbol{x} := [x_1, \dots, x_N]^\top \in \mathbb{R}^N$, $\boldsymbol{y} := [y_1^\top, \dots, y_N^\top]^\top \in \mathbb{R}^{2N}$, $\boldsymbol{w} := [\boldsymbol{w}_1^\top, \dots, \boldsymbol{w}_N^\top]^\top \in \mathbb{R}^{2N}$. As so, the measurement model for each node $i \in \{1, \dots, N\}$ can be written as:

$$\boldsymbol{y}_i = \sum_{j=1}^{N} A_{ij}\boldsymbol{x}_j + \boldsymbol{w}_i = \sum_{j \in \mathcal{N}_i^+} A_{ij}\boldsymbol{x}_j + \boldsymbol{w}_i \qquad (A_{ij} = 0 \text{ if } j \notin \mathcal{N}_i^+)\,,$$

where A_{ij} can be obtained from matrix $\begin{bmatrix} I_{N_b} & L^\top \end{bmatrix}^\top$ after a row and column permutation. The overall measurement model can be rewritten as

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{w}\,,$$

where $A := [A_1^\top, \dots, A_N^\top]^\top \in \mathbb{R}^{2N \times N}$ and $A_i := [A_{i1}, \dots, A_{iN}] \in \mathbb{R}^{2 \times N}$.

Now that the measurement model has been derived, it is possible to show why, even though the local costs are locally coupled, each agents needs information coming from two-hops neighbors. The cost function used is the quadratic one, which allows an easy derivation of the algorithm. However, the two-hops information dependence appears in more general functions, and in particular in the one used in the simulation section.

One of the standard estimation technique is to minimize the 2-norm of the residuals. According to this estimation, the cost functions employed are:

$$J_i(x_i, \{x_j\}_{j \in \mathcal{N}_i}) := \frac{1}{2}\|\boldsymbol{y}_i - A_i\boldsymbol{x}\|^2, \quad J(\boldsymbol{x}) := \sum_{i=1}^{N} J_i(x_i, \{x_j\}_{j \in \mathcal{N}_i}) = \frac{1}{2}\|\boldsymbol{y} - A\boldsymbol{x}\|^2,$$

whose gradient and Hessians are

$$\nabla J(\boldsymbol{x}) = A^\top(A\boldsymbol{x} - \boldsymbol{y})\,, \qquad \nabla^2 J(\boldsymbol{x}) = H = A^\top A\,,$$
$$H_{ij} = \sum_{\ell=1}^{N} A_{\ell i}^\top A_{\ell j} = \sum_{\ell \in \mathcal{N}_i^+} A_{\ell i}^\top A_{\ell j} = \sum_{\ell \in (\mathcal{N}_i^+ \cap \mathcal{N}_j^+)} A_{\ell i}^\top A_{\ell j}.$$

The optimal (centralized) least squares solution[1] is given by:

$$\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{x}} J(\boldsymbol{x}) = (A^\top A)^{-1} A^\top \boldsymbol{y}\,.$$

A standard approach to asymptotically obtain the optimal solution (in a centralized approach) is to employ an iterative algorithm based on the generalized gradient descent (in order to simplify the notation, the variables at the current time step are denoted without the time, and the variables at the following time step are denoted with a $+$ superscript):

$$\boldsymbol{x}^+ = \boldsymbol{x} - \epsilon D^{-1} A^\top (A\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{x} - \epsilon D^{-1} \nabla J(\boldsymbol{x}) = \boldsymbol{x} - \epsilon D^{-1}(H\boldsymbol{x} - A^\top \boldsymbol{y})\,,$$

where ϵ is a suitable stepsize and D is a strictly positive definite matrix, i.e. $D > 0$. Note that the Hessian of J is a strictly positive matrix so it is possible to substitute D with the Hessian, recovering the Newton method. A typical way to solve the previous update in a distributed fashion is to pick a block-diagonal matrix D, i.e. $D = \mathrm{blkdiag}(D_1, \ldots, D_N)$, so that the previous centralized update can be written as

$$\begin{aligned} x_i^+ &= x_i - \epsilon D_i^{-1}\Big(\sum_{j=1}^{N} H_{ij} x_j - \sum_{j=1}^{N} A_{ji}^\top \boldsymbol{y}_j\Big) \\ &= x_i - \epsilon D_i^{-1}\Big(\sum_{j=1}^{N}\sum_{\ell=1}^{N} A_{\ell i}^\top A_{\ell j} x_j - \sum_{j=1}^{N} A_{ji}^\top \boldsymbol{y}_j\Big) \\ &= x_i - \epsilon D_i^{-1}\left(\sum_{j \in \mathcal{N}_\ell^+, \forall \ell \in \mathcal{N}_i^+} A_{\ell i}^\top A_{\ell j} x_j - \sum_{j \in \mathcal{N}_i^+} A_{ji}^\top \boldsymbol{y}_j\right), \end{aligned} \tag{4.3}$$

where in the last step the property that $A_{ij} = 0$ if $j \notin \mathcal{N}_i^+$ was exploited. Note that the term within brackets concerns the gradient evaluation (and so it has to be evaluated also when a sub-gradient method is used). While the second summation involves only

[1] The formulation can be extended to the weighed least square solutions if noise with different variances R are included which would lead to the solution $\boldsymbol{x}^* = (A^\top R^{-1} A)^{-1} A^\top R^{-1} \boldsymbol{y}$, but for the sake of clarity in the notation of this section, it is omitted.

measurements that belong to the neighbors of node i, the first summation requires the node i to collect the state variables x_j that belongs to the neighbors of the neighbors. As so, this implementation is not really distributed, since two-hop communication is required. Although this is not impossible from a practical perspective, it requires substantial additional communication and synchronization efforts. An alternative approach that allows the implementation of a truly distributed algorithm is to create the following additional local variable at each node i:

$$\boldsymbol{z}_i = A_i \boldsymbol{x} = \sum_{j \in \mathcal{N}_i^+} A_{ij} x_j \,, \quad \forall i \,,$$

which can be collected in the vector $\boldsymbol{z} := [\boldsymbol{z}_1^\top, \dots, \boldsymbol{z}_N^\top]^\top$, so that in matrix form the previous expression can be written as $\boldsymbol{z} = A\boldsymbol{x}$. With this notation the generalized gradient descent can be written as:

$$\begin{aligned} \boldsymbol{z}_i^+ &= \sum_{j \in \mathcal{N}_i^+} A_{ij} x_j \\ x_i^+ &= x_i - \epsilon D_i^{-1} \Big(\sum_{\ell=1}^{N} A_{\ell i}^\top \underbrace{\sum_{j=1}^{N} A_{\ell j} x_j}_{\boldsymbol{z}_\ell} - \sum_{j=1}^{N} A_{ji}^\top \boldsymbol{y}_j \Big) \\ &= x_i - \epsilon D_i^{-1} \sum_{\ell \in \mathcal{N}_i^+} A_{\ell i}^\top (\boldsymbol{z}_\ell^+ - \boldsymbol{y}_\ell) \,. \end{aligned}$$

This alternative solution requires two communication rounds to compute x_i^+, since first it is necessary to send the x_is to compute $\boldsymbol{z}_i^+$s, and then to transmit the $\boldsymbol{z}_i$s. For simplicity, here it is assumed that node i sends its own measurements $\boldsymbol{y}_i$ to its neighbors (which store it) at the initialization phase, since the measurements do not change during the course of the evolution of the algorithm. Again, a double communication exchange per iteration requires additional synchronization. In practical scenarios, such as using PLC protocols, synchronization of transmissions and updates can be difficult. Moreover packet losses might occur, i.e. some messages from the neighbors might not be received. A naive solution to both problems is to use local registers that keep in memory the latest message received from the neighbors, and then use these values whenever an update of the local variables x_is and $\boldsymbol{z}_i$s is needed. It can be shown that this is equivalent to a scenario where every node $j \in \mathcal{N}_i$ use a delayed version of the local variables x_is and $\boldsymbol{z}_i$s. Since the variables $\boldsymbol{z}_i$s are function of the (possibly delayed) state variables x_is, the update of the variables x_is can be rewritten as a function of the delayed version of the variables

x_is. More specifically, the previous update equations can be written as

$$\boldsymbol{z}_i(k+1) = \sum_{j\in\mathcal{N}_i^+} A_{ij}x_j(\tau'_{ij}(k))\,, \tag{4.4}$$

$$x_i(k+1) = x_i(k) - \epsilon D_i^{-1}\Big(\sum_{\ell=1}^{N} A_{\ell i}^\top \underbrace{\sum_{j=1}^{N} A_{\ell j}x_j(\tau_{\ell j}(k))}_{\boldsymbol{z}_\ell(\tilde{\tau}_{\ell j}(k))} - \sum_{j=1}^{N} A_{ji}^\top \boldsymbol{y}_j\Big)\,. \tag{4.5}$$

where $0 \leq \tau_{ij}(k), \tau'_{ij}(k) \leq k$ represent the delay with respect to the current time k according to which variable x_j appears in the update of $\boldsymbol{z}_i$ and x_i respectively, and $\tilde{\tau}_{\ell j}(k)$ is the delay according to which $\boldsymbol{z}_\ell$ appears in the update of x_i. All these delays depend on the specific sequence of packet losses and variable updates, and explicitly included the time dependency of each variable. Note that in the last equation a variable x_j might appear with multiple instances with different delays into the update of the variable x_i. As a consequence, it is not possible to write the variables' evolution of the original generalized gradient descent algorithm given in Eqn. (4.3) as a partially asynchronous iterative methods (see chapter 7 of Bertsekas and Tsitsiklis (1989)), that is as

$$x_i(k+1) = x_i(k) - \epsilon D_i^{-1}\Big(\sum_{j=1}^{N} H_{ij}x_j(\tau_{ij}(k)) - \sum_{j=1}^{N} A_{ji}^\top \boldsymbol{y}_j\Big)\,. \tag{4.6}$$

This implies that it is not possible to use the results of the extensive body of literature related to these methods.

Motivated by this observation, an alternative mathematical machinery based on Lyapunov theory and the separation of time scale principle will be proposed to prove convergence of the asynchronous algorithm (4.5) for a sufficiently small stepsize ϵ.

The ideas used to obtain the algorithm can also be applied to more general convex problems. For example, in the presence of outliers or sensor faults in order, more robust estimators than least squares should be used. In particular, outliers are measurements that are completely wrong.

A common way to enforce robustness in the estimation is to replace the quadratic cost function defined above with the 1-norm of the residuals, that is

$$J_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) = \|\boldsymbol{y}_i - A_i\boldsymbol{x}\|_1\,. \tag{4.7}$$

As can be seen from the left part of Figure 4.2, using a 1-norm, an outlier is weighted much less with respect to its weight using the 2-norm, and so it does not influence too much the estimate.

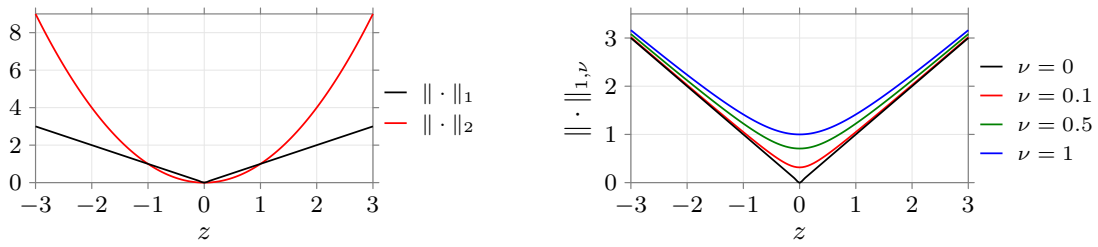


Figure 4.2: On the left comparison between the 2-norm and 1-norm. On the right, modified 1-norm for different values of the parametere ν.

However, since (4.7) is not differentiable, it cannot be directly used in the procedure just explained, since the gradient is needed. To deal with this issue, in the Simulation section 4.6, where this cost function is employed, the following modification of the 1-norm Argaez, Ramirez, and Sanchez (2011) is used

$$\|\cdot\|_{1,\nu} : \mathbb{R}^N \to \mathbb{R}\,, \quad \boldsymbol{x} \mapsto \|\boldsymbol{x}\|_{1,\nu} := \sum_{i=1}^{N} \sqrt{x_i^2 + \nu}\,, \tag{4.8}$$

where $\nu > 0$ is such that the smaller the selected value of ν is, the better the approximation of the 1-norm is (see the right part of Figure 4.2. In particular, the approximation of each term in the summation of the cost function is quadratic when x_i belongs to a small neighborhood of 0.

The next two sections generalize what was presented here, providing a fully distributed generalized gradient descent algorithm which is resilient to lossy communication.

4.4 Synchronous update and reliable communication

The idea is to first present an algorithm for the case of synchronous and ideal, i.e., reliable, communications among neighbors, and then to consider the extension to the more realistic case of unreliable communication in Section 4.5.

Consider the optimization Problem (4.1). In the ideal communication case, one possible choice to iteratively solve Problem (4.1) is to exploit the so called *generalized gradient descent* iteration

$$\boldsymbol{x}^+ = \boldsymbol{x} - \epsilon D^{-1}(\boldsymbol{x}) \nabla J(\boldsymbol{x})\,, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0\,, \tag{4.9}$$

where $\nabla J(\boldsymbol{x}) := \left[\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}}\right]^\top$ is the gradient of J evaluated at the current value $\boldsymbol{x}$, $D(\boldsymbol{x})$

is a generic positive definite matrix, possibly function of $\boldsymbol{x}$ itself, and ϵ a suitable positive constant, referred to as *step size*. Observe that depending on the particular choice of $D(\boldsymbol{x})$, Eq. (4.9) describes various types of algorithms. Indeed, if $D(\boldsymbol{x}) = I$, the standard gradient descent iteration is obtained; if $D(\boldsymbol{x})$ is chosen to be diagonal with diagonal elements equal to those of the Hessian matrix, then a Jacobi descent iteration is retrieved; while, if $D(\boldsymbol{x})$ is equal to the entire Hessian, then Eq. (4.9) returns classical Newton's iteration.
The algorithm proposed is inspired by a particular case of (4.9). Namely, $D(\boldsymbol{x})$ is chosen to be a *block diagonal* matrix such that

$$D(\boldsymbol{x}) = \text{blkdiag}(D_1(\boldsymbol{x}), \dots, D_N(\boldsymbol{x}))\,, \qquad D_i(\boldsymbol{x}) := \nabla^2_{ii} J(\boldsymbol{x})\,, \quad i \in \mathcal{V}\,, \tag{4.10}$$

i.e., where each diagonal block coincides with the second order derivative of J w.r.t. $\boldsymbol{x}_i$. This algorithm is denoted as *block Jacobi*.

Thanks to this choice for the matrix D, Eq. (4.9) can be split into partial state updates each of which equal to

$$\boldsymbol{x}_i^+ = \boldsymbol{x}_i - \epsilon D_i^{-1}(\boldsymbol{x}) \nabla_i J(\boldsymbol{x})\,, \quad i \in \mathcal{V}\,. \tag{4.11}$$

Now, it is convenient to explicitly take into account the separable structure of the cost function J in order to show that each gradient block $\nabla_i J$ as well as each D_i block can be computed exploiting only (sub-)local information coming from agent's i *two-steps neighbors*, i.e., agents connected to agent i by a directed path of length two. Indeed, for the gradient it holds that

$$\nabla_i J(\boldsymbol{x}) = \sum_{j \in \mathcal{N}_i^+} \nabla_i J_j(\{\boldsymbol{x}_\ell\}_{\ell \in \mathcal{N}_j^+}) = \nabla_i J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j \in \mathcal{N}_i}) + \sum_{j \in \mathcal{N}_i} \nabla_i J_j(\boldsymbol{x}_j, \{\boldsymbol{x}_\ell\}_{\ell \in \mathcal{N}_j}). \tag{4.12}$$

The first term on the right-hand side of Eq. (4.12) depends only on information coming from $j \in \mathcal{N}_i^+$; while, the second term possibly depends on information coming from neighbors of node i and from the neighbors of its neighbors, $\ell \in \mathcal{N}_j^+$. A similar reasoning applies to D_i. Indeed,

$$\begin{aligned} D_i(\boldsymbol{x}) &= \sum_{j \in \mathcal{N}_i^+} \nabla^2_{ii} J_j(\{\boldsymbol{x}_\ell\}_{\ell \in \mathcal{N}_j^+}) \\ &= \nabla^2_{ii} J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j \in \mathcal{N}_i}) + \sum_{j \in \mathcal{N}_i} \nabla^2_{ii} J_j(\boldsymbol{x}_j, \{\boldsymbol{x}_\ell\}_{\ell \in \mathcal{N}_j})\,. \end{aligned} \tag{4.13}$$

Again, the first term in the right-hand side of Eq. (4.13) depends only on node i direct neighbors, $j \in \mathcal{N}_i^+$, while the second term requires information coming from the neighbors of its neighbors. In view of a distributed computation, it is assumed that each agent $i \in \mathcal{V}$, once gathered the neighbors states $\{\boldsymbol{x}_j\}_{j\in\mathcal{N}_i}$, can compute and store in its local memory, in addition to the state $\boldsymbol{x}_i$, the following variables

$$\boldsymbol{\rho}_i^{(j)}(\boldsymbol{x}) := \nabla_j J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j\in\mathcal{N}_i}), \qquad \xi_i^{(j)}(\boldsymbol{x}) := \nabla_{jj}^2 J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j\in\mathcal{N}_i}), \tag{4.14}$$

which represent the partial components of the first and second derivatives of its local cost J_i with respect to variable $\boldsymbol{x}_j$, evaluated at the current state value. Observe that, since in a distributed framework each agent is assumed to have information only regarding its local cost J_i, the $\boldsymbol{\rho}$'s and ξ's variables represents the quantities that agent i must compute and send to its neighbors in order to let them compute their corresponding gradient and hessian blocks. Likewise, agent i needs to receive similar variables from each one of its neighbors. Indeed, thanks to Eqs. (4.12)–(4.13), it holds that

$$\nabla_i J(\boldsymbol{x}) = \sum_{j\in\mathcal{N}_i^+} \boldsymbol{\rho}_j^{(i)}(\boldsymbol{x}), \qquad D_i(\boldsymbol{x}) = \sum_{j\in\mathcal{N}_i^+} \xi_j^{(i)}(\boldsymbol{x}). \tag{4.15}$$

As stressed above, to iteratively compute (4.11), each agent $i \in \mathcal{V}$ can perform its computations autonomously assuming it has at its disposal information coming from its two-steps neighbors. However, this presents two major drawbacks:

1. it clashes with a truly distributed setting which exploits the exchange of information only among one-step neighbors;
2. within successive iterations, to ensure consistency and thus convergence of the procedure to a minimizer of Problem (4.1), all the communications must be synchronous and reliable.

To workaround the first issue one possible solution would be, at each iteration, to perform two communication rounds among one-step neighbors as illustratively shown in Figure 4.3. The first round is used to exchange the state values among neighboring agents in order them to compute all the partial information terms according to Eqs. (4.14)–(4.15); the second round is used to communicate the computed variables in order to perform the state update as in Eq. (4.11). Regarding the second issue, it necessarily enforces the use of suitable synchronization algorithms as well as re-transmission protocols in case of packet failures.

A more compact description of the procedure is given in in Algorithm 4.1 in which $\texttt{flag}_{\text{transmission}}$ denotes a variable to control communication and update among the agents.

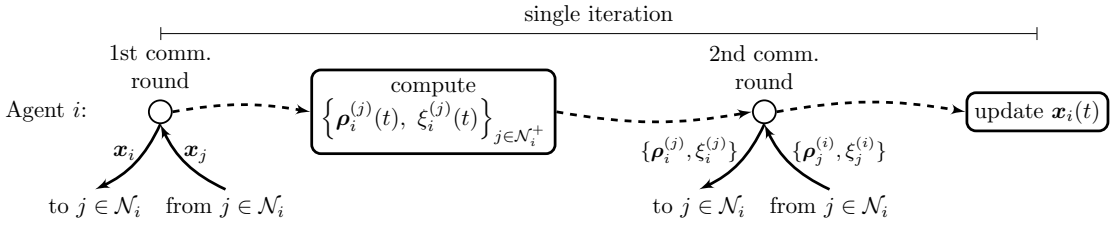


Figure 4.3: Communication scheme to perform one single block Jacobi iteration (4.11) in a distributed setting which assumes only information exchange among one-step neighbors.

Even though this procedure provides a possible solution to the problem, this is not really satisfactory for real-world applications. Consequently, next section presents a truly distributed and resilient iterative procedure which, by naturally exploiting information coming from one-step neighbors and being resilient to packet losses and communication non idealities, is much more appealing from an engineering perspective.

Algorithm 4.1 Distributed Block Jacobi algorithm (node i).

Require: $\boldsymbol{x}_i^o$, ϵ

$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i^o$

if $\texttt{flag}_{\texttt{transmission}} = 1$ **then**

  Broadcast: $\boldsymbol{x}_i$

  Receive: $\boldsymbol{x}_j, \quad \forall j \in \mathcal{N}_i$

  $\boldsymbol{\rho}_i^{(j)} \leftarrow \nabla_j J_i(\{\boldsymbol{x}_k\}_{k\in\mathcal{N}_j^+}), \quad \forall j \in \mathcal{N}_i^+$

  $\xi_i^{(j)} \leftarrow \nabla_{jj}^2 J_i(\{\boldsymbol{x}_k\}_{k\in\mathcal{N}_j^+}), \quad \forall j \in \mathcal{N}_i^+$

  Broadcast: $\boldsymbol{\rho}_i^{(j)}, \xi_i^{(j)}, \quad \forall j \in \mathcal{N}_i$

  Receive: $\{\boldsymbol{\rho}_j^{(i)}, \xi_j^{(i)}\}, \quad \forall j \in \mathcal{N}_i$

  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \epsilon\Big(\sum_{j\in\mathcal{N}_i^+} \xi_j^{(i)}\Big)^{-1}\Big(\sum_{j\in\mathcal{N}_i^+} \boldsymbol{\rho}_j^{(i)}\Big)$

end if

4.5 Asynchronous updates and unreliable communication: the Resilient Block Jacobi (RBJ) algorithm

In this section the assumption on ideal communication is relaxed. From now on communication is asynchronous and unreliable. As a consequence each agent might either receive asynchronous information coming from its neighbors, or not receive it. In particular, a modified iteration is presented and its corresponding iterative algorithm is analyzed. This new algorithm is referred to as *resilient block Jacobi*. It (i) exploits only information coming from one-step neighbors; (ii) requires only one communication round per algorith-

mic iteration; (iii) is based on an asynchronous communication protocol; (iv) is resilient to communication failures. The algorithm is first presented for the general case of separable convex costs. Later, it is particularized to suit two special cases to show its flexibility.

Consider the standard block Jacobi iteration (4.11). As pointed out in Section 4.4, the procedure exhibits some fundamental issues which deeply compromise its distributed and asynchronous implementation and also its robustness properties. Thus, it is necessary to suitably modify iteration (4.11) in order to obtain an algorithm more appropriate for a real distributed application.

The proposed modification is apparently naive since the idea is to simply equip each agent with an additional amount of memory storage to keep track of the last received and available information corresponding to each neighbor. This additional memory is then used to perform Eq. (4.11). Indeed, note that in the block-Jacobi algorithm, if agent i does not receive some of the information coming from its neighbors, it does not have the necessary information to synchronously compute neither (4.14) nor (4.15) and thus it is not able to update its state according to (4.11).
To model randomly occurring packet losses it is convenient to introduce the indicator function

$$\gamma_j^{(i)}(k) = \begin{cases} 1 & \text{if } i \text{ received the information sent by } j \text{ at iteration } k \\ 0 & \text{otherwise.} \end{cases}$$

with the assumption that $\gamma_i^{(i)}(k) = 1$, since node i has always access to its local variables. Then, as suggested above, the main idea is to equip each agent i with the auxiliary variables $\left\{\widehat{\boldsymbol{x}}_j^{(i)}, \widehat{\boldsymbol{\rho}}_j^{(i)}, \widehat{\xi}_j^{(i)}\right\}_{j\in\mathcal{N}_i}$, used to keep track of the last available information received by i from each of its neighbors. Specifically, the dynamic for the j-th set of additional variables is given by

$$\left\{\widehat{\boldsymbol{x}}_j^{(i)}(k), \widehat{\boldsymbol{\rho}}_j^{(i)}(k), \widehat{\xi}_j^{(i)}(k)\right\} = \begin{cases} \left\{\boldsymbol{x}_j(k), \boldsymbol{\rho}_j^{(i)}(k), \xi_j^{(i)}(k)\right\}, & \text{if } \gamma_j^{(i)}(k) = 1\,; \\ \left\{\widehat{\boldsymbol{x}}_j^{(i)}(k-1), \widehat{\boldsymbol{\rho}}_j^{(i)}(k-1), \widehat{\xi}_j^{(i)}(k-1)\right\}, & \text{if } \gamma_j^{(i)}(k) = 0\,. \end{cases} \tag{4.16}$$

Thanks to this additional memory, at every algorithmic iteration, each agent can perform its local update which is inspired on Eq. (4.11):

$$\boldsymbol{x}_i(k+1) = \boldsymbol{x}_i(k) - \epsilon \left(\sum_{j\in\mathcal{N}_i^+} \widehat{\xi}_j^{(i)}(k)\right)^{-1} \left(\sum_{j\in\mathcal{N}_i^+} \widehat{\boldsymbol{\rho}}_j^{(i)}(k)\right). \tag{4.17}$$

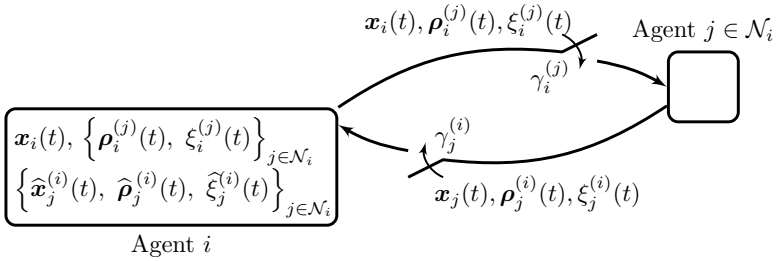


Figure 4.4: Memory storage and communication scheme between pairs of neighbors agents for the RBJ algorithm.

The differences between Eqs. (4.11) and (4.17) are mainly two:

1. the variables in agent i's memory used to store the first and second partial derivatives of J_i w.r.t. $\boldsymbol{x}_j$, $j \in \mathcal{N}_i$, are necessarily computed as

$$\boldsymbol{\rho}_i^{(j)}(k) = \nabla_j J_i(\boldsymbol{x}_i(k), \{\widehat{\boldsymbol{x}}_\ell^{(i)}(k)\}_{\ell\in\mathcal{N}_i}), \quad \xi_i^{(j)}(k) = \nabla_{jj}^2 J_i(\boldsymbol{x}_i(k), \{\widehat{\boldsymbol{x}}_\ell^{(i)}(k)\}_{\ell\in\mathcal{N}_i}), \tag{4.18}$$

   that is, they are evaluated at the last stored states' values; likewise, the values of the additional variables $\{\widehat{\boldsymbol{\rho}}_j^{(i)},\ \widehat{\xi}_j^{(i)}\}_{j\in\mathcal{N}_i}$ correspond to those last received from each neighbor and computed by each of them using the last available information on their neighbors' states;

2. conversely to the synchronous implementation of the algorithm, at each iteration only one communication round is performed. This means that the agents send only one packet per iteration, consisting of the state and the partial derivatives. See Figure 4.4 for an illustrative representation.

Thanks to this simple modification the agents can perform their updates asynchronously and independently. Moreover, since only one communication round per iteration is required, both the communication burden and the number of possible communication failures are reduced. Nevertheless, it is worth stressing that, even if no packet losses occur, the classical block Jacobi and our resilient block Jacobi iteration does not exactly coincide. Indeed, in the resilient case, by sending only one packet per iteration, the state and the partial derivative information would be "delayed" one from each other of one iteration if compared with the synchronous implementation. The *resilient block Jacobi* algorithm (hereafter referred to as RBJ algorithm) for separable convex functions is formally described in Algorithm 4.2 where it is presented in an event-based update performed by a generic node i. The variables $\texttt{flag}_{\texttt{transmission}}, \texttt{flag}_{\texttt{reception}}, \texttt{flag}_{\texttt{update}}$ are flag variables which determines which specific action a node is performing, namely

transmission, reception or update. When each action is started it cannot be interrupted, but the specific order or consecutive calls of an action do not impair the convergence of the proposed algorithm and therefore the algorithm can be used independently of the specific communication protocol or CPU multitasking scheduling (if some assumptions later introduced on the communication are met).

Remark 4.5.1. The memory (and consequently also the communication) requirement of the RBJ can be demanding. In fact, assuming for simplicity that all the variables $\boldsymbol{x}_i$ have the same dimension $\bar{n}$, node i has to keep in memory (from one iteration to the following one) $2|\mathcal{N}_i|+1$ vectors of dimension $\bar{n}$ (namely for $\boldsymbol{x}_i, \{\widehat{\boldsymbol{x}}_j\}_{j\in\mathcal{N}_i}$ and $\left\{\widehat{\boldsymbol{\rho}}_j^{(i)}\right\}_{j\in\mathcal{N}_i}$), and $|\mathcal{N}_i|$ matrices of dimension $\bar{n}\times\bar{n}$ (for $\left\{\widehat{\xi}_j^{(i)}\right\}_{j\in\mathcal{N}_i}$). If memory, communication and computational complexity are a concern, it is possible to modify the proposed algorithm mimicking the standard gradient descent algorithm. In this framework, the second order information is not needed and therefore the variables $\xi_j^{(i)}, \widehat{\xi}_j^{(i)}$ (`lines 5, 6, 20` in Algorithm 4.2) do not need to be computed and stored (saving the memory space of $|\mathcal{N}_i|$ matrices of dimension $\bar{n}\times\bar{n}$). The update for the local variable $\boldsymbol{x}_i$ (`line 22` in Algorithm 4.2) is replaced with the following:

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \epsilon \sum_{j\in\mathcal{N}_i^+} \widehat{\boldsymbol{\rho}}_j^{(i)} .$$

Obviously, the price to pay for this choice is a likely decrease in convergence speed. This robust and asynchronous version of the gradient descent algorithm is denoted as *resilient gradient descent* (RGD) algorithm.

Remark 4.5.2. If the local cost functions are quadratic, i.e:

$$J_i(\boldsymbol{x}_i, \{\boldsymbol{x}_j\}_{j\in\mathcal{N}_i}) = \frac{1}{2}\|\boldsymbol{y}_i - A_i\boldsymbol{x}\|_{W_i}^2 = \frac{1}{2}(\boldsymbol{y}_i - \sum_{j\in\mathcal{N}_i^+} A_{ij}\boldsymbol{x}_j)^\top W_i(\boldsymbol{y}_i - \sum_{j\in\mathcal{N}_i^+} A_{ij}\boldsymbol{x}_j) ,$$

where $W_i > 0$ are the local weights, then the problem to be solved becomes a Weighted Least Squares problem. For this special case, the gradient and the hessian components simplify to:

$$\boldsymbol{\rho}_i^{(j)}(x) := A_{ij}^\top W_i(\sum_{j\in\mathcal{N}_i^+} A_{ij}\boldsymbol{x}_j - \boldsymbol{y}_i) , \qquad \xi_i^{(j)}(x) := A_{ij}^\top W_i A_{ij} , \tag{4.19}$$

therefore the RBJ Algorithm can be simplified by substituting `lines 10` and `11` with

Algorithm 4.2 Resilient Block Jacobi (RBJ) Algorithm (node i)

Require: $\boldsymbol{x}_i^o$, ϵ

Initialization (atomic)

$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i^o$

$\widehat{\boldsymbol{x}}_j^{(i)} \leftarrow \mathbf{0}, \quad \forall j \in \mathcal{N}_i$

$\boldsymbol{\rho}_i^{(j)} \leftarrow \mathbf{0}, \quad \forall j \in \mathcal{N}_i$

$\widehat{\boldsymbol{\rho}}_j^{(i)} \leftarrow \mathbf{0}, \quad \forall j \in \mathcal{N}_i$

$\xi_i^{(j)} \leftarrow I_{n_j}, \quad \forall j \in \mathcal{N}_i$

$\widehat{\xi}_j^{(i)} \leftarrow I_{n_i}, \quad \forall j \in \mathcal{N}_i$

$\texttt{flag}_{\texttt{transmission}} \leftarrow 1$ (optional)

Transmission (atomic)

if $\texttt{flag}_{\texttt{transmission}} = 1$ **then**

  $\texttt{transmitter\_node\_ID} \leftarrow i$

  $\boldsymbol{\rho}_i^{(j)} \leftarrow \nabla_j J_i(\boldsymbol{x}_i, \{\widehat{\boldsymbol{x}}_\ell^{(i)}\}_{\ell \in \mathcal{N}_i}), \quad \forall j \in \mathcal{N}_i$

  $\xi_i^{(j)} \leftarrow \nabla_{jj}^2 J_i(\boldsymbol{x}_i, \{\widehat{\boldsymbol{x}}_\ell^{(i)}\}_{\ell \in \mathcal{N}_i}), \quad \forall j \in \mathcal{N}_i$

  Broadcast: $\texttt{transmitter\_node\_ID}, \boldsymbol{x}_i, \{\boldsymbol{\rho}_i^{(j)}, \xi_i^{(j)}\}_{j \in \mathcal{N}_i}$

  $\texttt{flag}_{\texttt{transmission}} \leftarrow 0$

  $\texttt{flag}_{\texttt{reception}} \leftarrow 1$ (optional)

end if

Reception (atomic)

if $\texttt{flag}_{\texttt{reception}} = 1$ **then**

  $j \leftarrow \texttt{transmitter\_node\_ID}$

  $\widehat{\boldsymbol{x}}_j^{(i)} \leftarrow \boldsymbol{x}_j$

  $\widehat{\boldsymbol{\rho}}_j^{(i)} \leftarrow \boldsymbol{\rho}_j^{(i)}$

  $\widehat{\xi}_j^{(i)} \leftarrow \xi_j^{(i)}$

  $\texttt{flag}_{\texttt{reception}} \leftarrow 0$

  $\texttt{flag}_{\texttt{update}} \leftarrow 1$ (optional)

end if

Estimate update (atomic)

if $\texttt{flag}_{\texttt{update}} = 1$ **then**

  $\widehat{\boldsymbol{\rho}}_i^{(i)} \leftarrow \nabla_i J_i(\boldsymbol{x}_i, \{\widehat{\boldsymbol{x}}_\ell^{(i)}\}_{\ell \in \mathcal{N}_i})$

  $\widehat{\xi}_i^{(i)} \leftarrow \nabla_{ii}^2 J_i(\boldsymbol{x}_i, \{\widehat{\boldsymbol{x}}_\ell^{(i)}\}_{\ell \in \mathcal{N}_i})$

  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \epsilon\Big(\sum_{j \in \mathcal{N}_i^+} \widehat{\xi}_j^{(i)}\Big)^{-1}\Big(\sum_{j \in \mathcal{N}_i^+} \widehat{\boldsymbol{\rho}}_j^{(i)}\Big)$

  $\texttt{flag}_{\texttt{update}} \leftarrow 0$

  $\texttt{flag}_{\texttt{transmission}} \leftarrow 1$ (optional)

end if

the following updates:

$$\boldsymbol{\rho}_i^{(j)} \leftarrow A_{ij}^\top W_i (A_{ii}\boldsymbol{x}_i + \sum_{j\in\mathcal{N}_i} A_{ij}\widehat{\boldsymbol{x}}_j^{(i)} - \boldsymbol{y}_i), \quad \forall j \in \mathcal{N}_i\,, \tag{4.20}$$

$$\xi_i^{(j)} \leftarrow A_{ij}^\top W_i A_{ij}\,. \tag{4.21}$$

It is clear from the previous expression, that the algorithm could be modified by having a preliminary phase when the $\xi_i^{(j)}$ are transmitted reliably to the neighbours so that eventually $\widehat{\xi}_i^{(j)} = \xi_i^{(j)}$, and then the algorithm could simply transmit the variables $\boldsymbol{x}_i, \boldsymbol{\rho}_i^{(j)}$ and update the variables $\boldsymbol{x}_i, \widehat{\boldsymbol{x}}_j^{(i)}, \widehat{\boldsymbol{\rho}}_j^{(i)}$ which are the only variables that evolve over time, thus considerably reducing the communication complexity which corresponds with that of the RGD algorithm. This specialized version of the RBJ is hereafter denoted as *resilient weighted least squares* (RWLS) algorithm.

Theoretical analysis of RBJ algorithm

To state the major theoretical result characterizing the convergence properties of the proposed RBJ algorithm, it is necessary to establish some properties for the asynchronous and lossy communication considered. As was done in the previous chapter, the assumptions needed on the communication are the ones introduced in Section 2.3, that is Assumptions 2.3.1 and 2.3.2. As a consequence of the combination of these assumptions, each agent $i \in \mathcal{V}$ receives information coming from each agent $j \in \mathcal{N}_i$ at least once within any window of $T = h\tau$ iterations of the algorithm.

It is now possible to state the following theorem:

Theorem 4.5.3 (Local convergence of the RBJ algorithm). *Let Assumptions 4.2.2, 2.3.1 and 2.3.2 hold. Moreover assume that the cost functions J_i are three-times differentiable and continuous. Consider Problem (4.1) and the RBJ algorithm. Let $\boldsymbol{x}^*$ be the minimizer of (4.1). There exists $\bar{\epsilon} > 0$ and $\delta > 0$, such that, if $0 < \epsilon < \bar{\epsilon}$ and $\|\boldsymbol{x}(0) - \boldsymbol{x}^*\| < \delta$, then the trajectory $\boldsymbol{x}(k)$, generated by the RBJ algorithm, converges exponentially fast to $\boldsymbol{x}^*$, i.e.,*

$$\|\boldsymbol{x}(k) - \boldsymbol{x}^*\| \leq C\rho^k$$

for some constants $C > 0$ and $0 < \rho < 1$.

The proof of Theorem 4.5.3 can be found in Appendix C, and basically relies on the separation of time scales principle between the dynamics of the states $\boldsymbol{x}_i$s and those of the auxiliary variables $\widehat{\boldsymbol{x}}_j^{(i)}$s, $\boldsymbol{\rho}_j^{(i)}$s, $\widehat{\boldsymbol{\rho}}_i^{(j)}$s, $\xi_j^{(i)}$s and $\widehat{\xi}_i^{(j)}$'s. Loosely speaking, the result builds on the idea that if the step-size ϵ is small enough, the variation of the true states

$\boldsymbol{x}_i$s is sufficiently slow and, despite the lossy communication, the values of the auxiliary variables stored in memory equal the true values.

Remark 4.5.4. The same argument used in the previous theorem can be applied to the *robust gradient descent* algorithm presented in Remark 4.5.1, under the weaker assumption that the cost functions J_i are two-times differentiable, thus providing the same local exponential convergence. Typically, the critical value $\bar{\epsilon}$ for the RGD algorithm is smaller than that of the RBJ algorithm, and consequently also the rate of convergence is slower.

Lemma 4.5.5 (Theorem 4 in Todescato et al. (2015))**.** *Let Assumptions 4.2.2, 2.3.1 and 2.3.2 hold. Consider Problem (4.1) with a quadratic cost function $J(\boldsymbol{x})$ and the RWLS algorithm. There exists $\bar{\epsilon}$ such that, if $0 < \epsilon < \bar{\epsilon}$, then, for any $\boldsymbol{x}(0) \in \mathbb{R}^n$, the trajectory $\boldsymbol{x}(k)$, generated by the RWLS algorithm, converges exponentially fast to the minimizer $\boldsymbol{x}^*$ of the corresponding problem, i.e.,*

$$\|\boldsymbol{x}(k) - \boldsymbol{x}^*\| \leq C\rho^k$$

for some constants $C > 0$ and $0 < \rho < 1$.

4.6 Smart power grid application

This section contains some simulative results obtained using the RBJ algorithm. The simulations involve the IEEE 123 nodes distribution grid benchmark (see Bolognani et al. (2014)). The problem addressed is the robust estimation of the voltage level at each node of the grid (except the PCC node which is assumed fixed and known), given voltage and current measurements in the presence of measurements outliers. Voltages and currents in an AC power distribution grid are complex values. In view of the state estimation problem considered, it is convenient to exploit an equivalent standard reformulation in rectangular coordinates. In particular, given the complex vectors of voltages and currents, denoted as $\mathfrak{v} \in \mathbb{C}^{122}$ and $\mathfrak{i}^c \in \mathbb{C}^{122}$ respectively, and the weighted Laplacian matrix $\mathfrak{L} \in \mathbb{C}^{122\times 122}$ describing the electric grid, thanks to Kirchhoff's voltage and current laws, it holds that

$$\mathfrak{i}^c = \mathfrak{L}\mathfrak{v}. \tag{4.22}$$

By rewriting voltages and currents in rectangular coordinates as

$$\boldsymbol{v} := [\Re(\mathfrak{v})^\top \ \Im(\mathfrak{v})^\top]^\top \in \mathbb{R}^{244}, \quad \boldsymbol{i}^c := [\Re(\mathfrak{i}^c)^\top \ \Im(\mathfrak{i}^c)^\top]^\top \in \mathbb{R}^{244}.$$

and, similarly, by splitting $\mathfrak{L}$ into its real and imaginary parts as

$$L = \begin{bmatrix} \Re(\mathfrak{L}) & -\Im(\mathfrak{L}) \\ \Im(\mathfrak{L}) & \Re(\mathfrak{L}) \end{bmatrix},$$

Eq. (4.22) is equivalent to

$$\boldsymbol{i}^c = L\boldsymbol{v}\,.$$

Thus, by assuming to collect both current and voltage measurements directly in rectangular coordinates[2], the measurement model reads as

$$\begin{bmatrix} \boldsymbol{y}^v \\ \boldsymbol{y}^{i^c} \end{bmatrix} = \begin{bmatrix} I \\ L \end{bmatrix} \boldsymbol{v} + \begin{bmatrix} \boldsymbol{w}^v \\ \boldsymbol{w}^{i^c} \end{bmatrix} + \begin{bmatrix} \boldsymbol{o}^v \\ \boldsymbol{o}^{i^c} \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{w}^v \\ \boldsymbol{w}^{i^c} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \sigma_v^2\,\mathrm{diag}(|\boldsymbol{v}|) & \\ & \sigma_{i^c}^2\,\mathrm{diag}(|\boldsymbol{i}^c|) \end{bmatrix} \right),$$

where $I \in \mathbb{R}^{244\times 244}$ is the identity matrix, $\boldsymbol{y}^v, \boldsymbol{y}^{i^c} \in \mathbb{R}^{244}$ are the measurements, collected in vector $\boldsymbol{y} \in \mathbb{R}^{488}$, $\boldsymbol{w}^v, \boldsymbol{w}^{i_c} \in \mathbb{R}^{244}$ are the measurements' noise, and $\boldsymbol{o}^v, \boldsymbol{o}^{i^c} \in \mathbb{R}^{244}$ are sparse vectors which contain possible measurement outliers. The standard deviation of the measurement errors is chosen as[3] $\sigma_v = 10^{-3}$[p.u.] and $\sigma_{i^c} = 10^{-1}$[p.u.]. Finally, concerning the outliers, 10% of the measurements are corrupted, and the distribution of the outliers is uniform between 1/100 and 1/80 of the respective measurement for voltages and between 1/2 and 1 of the respective current measurement.
As suggested at the end of Section 4.3, to perform robust state estimation in the presence of outliers, one interesting choice for the cost function is the modified 1-norm defined is Eq. (4.8) as

$$\|\boldsymbol{r}\|_{1,\nu}$$

where $\boldsymbol{r} = \boldsymbol{y} - A\boldsymbol{v}$ are the measurements residuals with $A = [I \;\; L^\top]^\top$. To run the RBJ algorithm the grid has to be partitioned. To do so, the feeder is divided into N non overlapping areas, and a computing unit, which can collect the measurements of the busses belonging to the area and can run the algorithm, is associated to each area. An example of the division in areas is given in Figure 4.5. The communication graph $\mathcal{G}$ can be obtained from the division in areas, and in particular, two units can communicate with each other if the two areas are physically connected (that is if there exist two busses,

[2] According to the future smart grids paradigm, it is assumed that each node of the grid is equipped with a smart measurement units, e.g., a Phasor Measurement Unit (PMU), which can return measurements of current and voltage. Usually, electric quantities are measured in polar coordinates. However, for the sake of simplicity, measurements are taken to be directly in rectangular coordinates, stressing that, thanks to a suitable linearization, it is always possible to pass from polar to rectangular coordinates.

[3] The choice for the measurements error standard deviations is dictated by the fact that the de facto standard for modern PMUs requires at most a 0.1% error in the voltage measurements. This translates in a current error of more or less 10%.

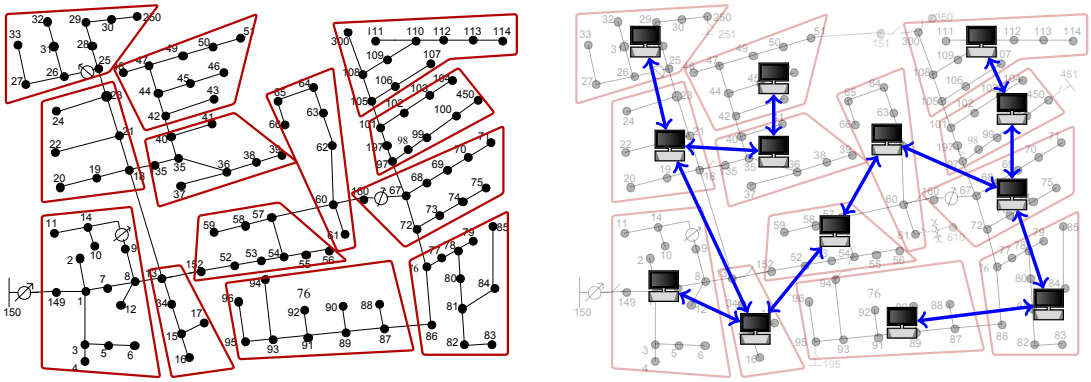


Figure 4.5: On the left, how the IEEE 123 distribution grid is divided in $N = 13$ areas. On the right, computing units associated to the areas, which can communicate only if an electric line is shared by busses belonging to the two respective areas.

each one belonging to one of the areas, which are connected by an electric wire).

The vectors $\boldsymbol{y}, \boldsymbol{x}$ and $\boldsymbol{r}$ and matrix A are divided according to the partition of the nodes in areas. Given the cost function, the values of $\boldsymbol{\rho}_j^{(i)}$ and $\xi_j^{(i)}$ are computed as:

$$\boldsymbol{\rho}_j^{(i)} = A_{ji}^\top \left((\mathrm{diag}(\boldsymbol{r}_j))^2 + \nu I \right)^{-1/2} \boldsymbol{r}_j, \quad \forall\, j \in \mathcal{N}_i^+$$
$$\xi_j^{(i)} = \nu A_{ji}^\top \left((\mathrm{diag}(\boldsymbol{r}_j))^2 + \nu I \right)^{-3/2} A_{ji}, \quad \forall\, j \in \mathcal{N}_i^+.$$

Note that these quantities are evaluated by agent j and then sent to agent i.

The RBJ algorithm is tested under two different scenarios, to evaluate the influence of different parameters involved in the algorithm. All the results showed are obtained averaging over 100 Monte Carlo runs (MCR).

In the first considered scenario, the influence of the number N of areas on the performance of the algorithm is shown. Observe that, for the case $N = 1$ the proposed RBJ algorithm resembles a Newton-Raphson iteration. Thus, in general and as shown in Figure 4.6, the fewer the number of areas, the faster the convergence rate. However, using fewer areas implies a bigger memory and communication burden (as long as N is different from 1, since $N = 1$ corresponds to the centralized scenario, where only the current state has to be stored from one iteration to the following one). Table 4.1 shows how the memory burden changes with N.

The second scenario analyzes the influence of the step size ϵ on the convergence rate. As can be seen from Figure 4.7, the convergence rate improves for bigger values of ϵ. Nevertheless, it is important to highlight the fact that the algorithm may diverge if the selected ϵ is too large.

N	1	3	7	13	28	123
Minimum # of elements	244	5110	1118	314	58	10
Maximum # of elements	244	11094	3068	1466	512	34

Table 4.1: For a given N, the table shows the minimum and maximum memory requirement (in terms of number of elements to be stored) needed to run the RBJ algorithm. In particular, it shows the memory required by the node which has the smaller memory burden, and the one required by the node with the highest memory burden.

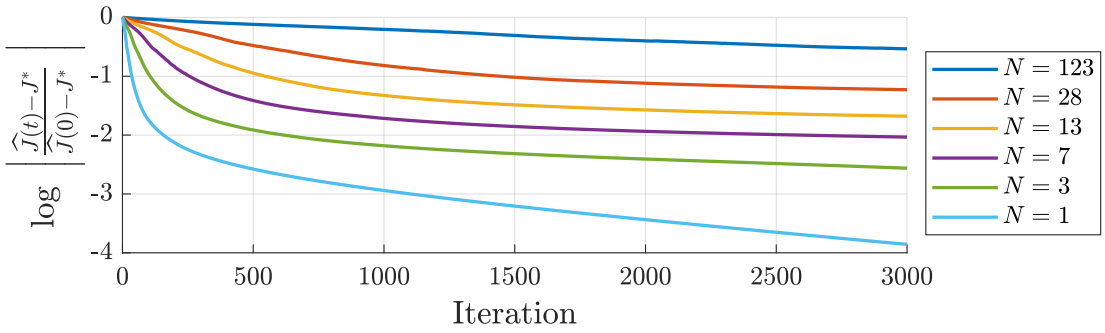


Figure 4.6: Normalized cost function as a function of the iteration, for different numbers of areas N. In all simulations there is a packet loss probability of 30% and $\epsilon = 0.0004$. The results are obtained averaging over 100 MCR. The initial condition for the algorithm are selected as the noisy measurement of the voltage.

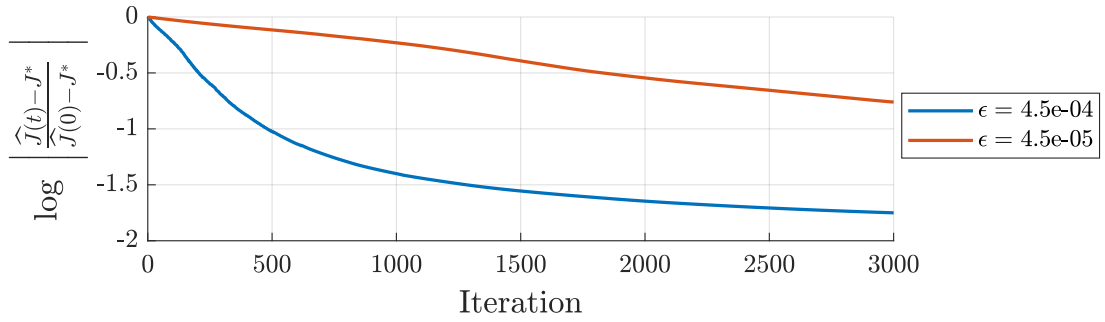


Figure 4.7: Normalized cost function as a function of the iteration, for different values of the parameter ϵ. The number of areas used is $N = 13$ and the results are obtained averaging over 100 MCR. In all simulations there is a packet loss probability of 30%. The initial condition for the algorithm are selected as the noisy measurement of the voltage.

The results show that the algorithm is robust to packet losses as can be seen both from Figures 4.6–4.7 where the algorithm has been tested considering a 30% packet loss probability. As last remark regarding the packet loss scenario, it is numerically observed that the higher the packet loss probability, the smaller the value of the step size to ensure convergence of the algorithm. Therefore, in the choice of the step size of the algorithm, which is still an open problem, the degree of reliability of the communication network

must be considered.

4.7 Final considerations on the minimization of locally coupled costs

This chapter focused on the minimization of the sum of locally coupled cost functions, under the assumption that the communication is unreliable and asynchronous. Particular emphasis is given to the communication effort required by the algorithm. In fact, the algorithm that is finally developed requires just one exchange for iteration, and this exchange comprises only information coming from one-hop neighbors. Moreover, each agent only estimates its own part of the optimization variable, and exchanges its part of the estimate only with the one-hop neighbors. This reduces to the minimum the sharing of information, and so this approach better protects the privacy of the information of each node, which is an important aspect (see Quinn (2009)). The algorithms in the literature either deal with a more general problem (the sum of functions with all the agents having to estimate the whole optimization variable) which requires a bigger communication effort since the locality of the problem is not exploited, or with a very similar problem but requiring synchronous communication. In all cases, packet losses are not considered in the literature, while here particular attention is given to the development of an algorithm working also in case of unreliable communication.

The algorithm proposed is based on the well-known Jacobi iteration. By leveraging Lyapunov theory and separation of time scale principle, robustness of the algorithm to packet drops and communication failures are proven. Some aspects of the algorithm need further research efforts. In particular it would be very interesting to be able to determine a priori the maximum value of the step size (namely $\bar{\epsilon}$). At the moment only its existence is proven, but its value is not known and the value of the step size has to be manually set until the algorithm converges. The setting of the step size is a difficult problem for almost all the optimization algorithms in the literature.

5

Average consensus and quadratic cost minimization

The contents of this chapter partly extend the following papers

Bof N., Carli R., and Schenato L. On the performance of consensus based versus Lagrangian based algorithms for quadratic cost functions. In *Proceedings of the 2016 European Control Conference*, pages 160–165. IEEE, 2016b

Bof N., Carli R., and Schenato L. Is ADMM always faster than average consensus? *Provisionally accepted on Automatica*, 2017d

Bof N., Carli R., and Schenato L. Average consensus with asynchronous updates and unreliable communication. In *Proceedings of the 20th IFAC World Congress, 2017*, volume 50, pages 601–606. IFAC, 2017a

The average consensus problem and the minimization of the sum of quadratic functions are tackled in this chapter. These two problems have a very strong relationship, and being able to solve one means being able to solve also the other. There are several applications for multi-agent systems in which either of these problems can be found, and this justify all the research effort devoted to this kind of problems.

A first part of the chapter gives a comparison on the convergence rates of well-known algorithms to solve either the consensus or the quadratic minimization problem. In

this comparison, particular attention is devoted to the influence of the communication graph, and specifically on its connectivity among agents. However, the communication is assumed synchronous and reliable.

Conversely, in the second part of the chapter the challenges of real-world communication are introduced. In particular, an algorithm for distributed average consensus working in an asynchronous and unreliable communication scenario is presented. This algorithm will be later used in Chapter 6.

5.1 Introduction and state of the art

This chapter deals with the average consensus problem in a multi-agent system and its relationship with the distributed minimization of quadratic functions. Namely, in an average consensus problem the agents in the system own a private quantity, and their goal is to evaluate the mean of the values of all these quantities. Average consensus, as will be shown in the following section, can be employed to solve the distributed minimization of the sum of quadratic functions, and vice-versa a consensus problem can be solved minimizing the sum of suitable quadratic cost function. As a consequence, there is again a strong relationship between this chapter and Chapter 6. However, since consensus and the quadratic case are really important and investigated problems, a whole chapter is devoted to these special cases. Moreover, the robust algorithm for average consensus developed in this chapter will be fundamental in the following one. In the remaining of this section, the term problem (if not specified) can refer to either of the problems (due to their strong connection).

This chapter's problem arise in several applications for multi-agent systems, e.g., in data fusion Xiao, Boyd, and Lall (2005), Bolognani, Favero, Schenato, and Varagnolo (2010), Garin and Schenato (2010) or clock synchronization Giridhar and Kumar (2006); Barooah and Hespanha (2007) for WSN, in sensors' localization problems Ravazzi, Frasca, Ishii, and Tempo (2013); Carron et al. (2014), in robot networks for flocking and coordination Blondel, Hendrickx, Olshevsky, and Tsitsiklis (2005); Jadbabaie, Lin, and Morse (2003); Nedić and Liu (2014) or for map building Carron, Todescato, Carli, Schenato, and Pillonetto (2015) and in state estimation of a power network Pasqualetti, Carli, and Bullo (2012). Moreover, there exist some algorithms which employ an average consensus algorithm as a building block, e.g. the Newton-Raphson Consensus for convex optimization (which will be introduced in Chapter 6), some distributed versions of the Kalman filter Cattivelli and Sayed (2010) or some algorithms for energy resources distribution in power grids Dominguez-Garcia and Hadjicostis (2010).

The first part of this chapter assumes an ideal communication network, that is synchronous and perfectly reliable. For such a communication scenario, a lot of solutions for this chapter's problem have been developed and also analyzed from different point of views. The works cited in the following will try to give an overview of the possible methods to adopt.

The average consensus algorithm, based on the use of stochastic matrices, has been widely studied, both in its standard form Garin and Schenato (2010); Boyd, Diaconis, and Xiao (2004); Olshevsky and Tsitsiklis (2009); Domínguez-García and Hadjicostis (2011), and in the accelerated one Olshevsky and Tsitsiklis (2009); Oreshkin, Coates, and Rabbat (2010); Muthukrishnan, Ghosh, and Schultz (1998); Liu and Morse (2011). These algorithms solve the average consensus problem.

Also primal sub-gradient methods can be employed to have the agents reach consensus. For example in Nedić et al. (2010) a sub-gradient methods is used to reach consensus in a constrained set-up (even though this consensus is not necessarily on the average). Also in this case stochastic matrices play an important role.

Lagrangian methods can be employed to solve the distributed minimization of quadratic functions. The Lagrangian approach which is used and studied the most to solve such problems is ADMM Shi, Ling, Yuan, Wu, and Yin (2014); Iutzeler, Bianchi, Ciblat, and Hachem (2016); Ling, Shi, Wu, and Ribeiro (2015); Makhdoumi and Ozdaglar (2016); Teixeira, Ghadimi, Shames, Sandberg, and Johansson (2013, 2016).

The aim of the first part of the chapter is to carry out a comparison on the convergence rates between consensus based algorithms (the standard consensus Garin and Schenato (2010) and the accelerated consensus Muthukrishnan et al. (1998)) on the one hand, and Lagrangian methods (the dual ascent algorithm Boyd et al. (2011) and ADMM Boyd et al. (2011)), on the other hand.

Concerning the ADMM, its analysis is carried out by rewriting it as a linear dynamical system as done in Erseghe, Zennaro, Dall'Anese, and Vangelista (2011). This latter paper considers the distributed minimization of the sum of quadratic cost functions for the special case in which this problem corresponds to an average consensus one. A comparison of the convergence rates of the consensus algorithm and ADMM is carried out, showing that ADMM is faster then the consensus algorithm for sparse graph. In this chapter, a more compact closed form expression for the rate of convergence of ADMM, using a different mathematical machinery than Erseghe et al. (2011), is given. Moreover, from a simulative point of view, more general (also multivariate) quadratic cost functions are analyzed, finding similar results. Differently from Shi et al. (2014) and Makhdoumi and Ozdaglar (2016), where, in the case of generic convex cost functions, upper bounds for

the ADMM convergence rate are offered, here the optimal convergence rate is given, upon restricting the cost functions to be quadratic and all with the same curvature. Moreover, the latter restriction, that allows Iutzeler et al. (2016) to optimize the distributed ADMM for ring communication graphs, is here exploited to optimize ADMM for general graphs (the ADMM presents a free parameter to be set, and in this sentence, optimize means to set this parameter in order to obtain the fastest possible version for the algorithm).

The main contribution of the first part of the chapter regards a study on the convergence rate for the different algorithms. In particular, this study shows how the latter is influenced by the graph connectivity and by the cost functions' curvatures. It is divided into two parts: firstly, the convergence rate of the different algorithms are analytically determined, assuming that the curvature of the cost functions are all equal. This analytical analysis shows that accelerated consensus Muthukrishnan et al. (1998) can be applied with very good results in all situations. Very interestingly, the calculations show also that while in consensus-based algorithms and in the dual ascent algorithm the convergence rate improves as the underlying graph gets more connected, in ADMM the convergence rate plateaus. Secondly, simulations done in more general scenarios show that these qualitative behaviors are almost always maintained, except for the dual ascent algorithm, whose performance highly deteriorates. The simulation part also points out that the main difference between the two types of algorithms is that the performance of the consensus-based algorithms is not influenced by the curvatures of the cost functions, differently from the Lagrangian-based algorithms. In fact, the curvatures strongly impact the rate of convergence of the Lagrangian methods, especially in the multivariate case. Moreover, both the consensus based algorithms and the Lagrangian based ones have some parameters to be set. However, the parameters for the consensus algorithms can be optimally chosen once the communication graph is given (even though to set them in an optimal way one has to employ a centralized approach), and the convergence rate remains the same for any curvatures of the cost functions. On the other hand, for the Lagrangian based algorithms, the optimal choice of the parameter depends on both the communication graph and the cost functions' curvatures, which implies that the tuning of this parameter is not determined only by the graph.

The second part of the chapter considers only the consensus problem (or the specific quadratic problem related to it), but the focus is on an asynchronous and unreliable communication scenario.

When unreliability in the communication is introduced, some works have adopted the acknowledge scheme Chen, Tron, Terzis, and Vidal (2010); Kar and Moura (2009, 2010) or assumed that each unit can determine whether the communication works Patterson

et al. (2007); Xiao et al. (2005). However, as already discussed, an acknowledge scheme has some disadvantages, and it might be preferable to develop a robust algorithm. Other interesting works related to consensus with imperfect communication are Aysal, Yildiz, Sarwate, and Scaglione (2009); Nedić (2011) (and with the last one having a wider applicability than the consensus problem). However, the approach sought for in this thesis is a deterministic one, while in these two works the approach is not so. The aim of the second part of the chapter is to find a robust and asynchronous algorithm to solve the consensus problem.

In an asynchronous setting, Bénézit, Blondel, Thiran, Tsitsiklis, and Vetterli (2010) introduce an algorithm that reaches average consensus using the so-called ratio consensus. A very interesting idea is introduced in Dominguez-Garcia, Hadjicostis, and Vaidya (2011) and Vaidya, Hadjicostis, and Dominguez-Garcia (2011), where the adopted communication is synchronous and unreliable. In these latter two works, a robust and synchronous algorithm inspired by Bénézit et al. (2010) is introduced.

Adopting the idea of mass transfer given in Vaidya et al. (2011), but using an asynchronous protocol as done in Bénézit et al. (2010), a new algorithm for average consensus is developed in Section 5.9. This algorithm is provably convergent to the average in an asynchronous and unreliable communication scenario. The convergence proof relies on the use of two assumptions concerning the communication scheme, one regarding the frequency of waking up of each node and the other regarding how many consecutive times a given link can fail. These two assumption (which are deterministic) allow to prove the exponential convergence of the algorithm, and this exponential property is really of interest here. As a matter of fact, the algorithm presented in the following chapter will utilize the algorithm for consensus developed in this chapter, and to prove convergence of the overall procedure the exponential convergence of the consensus algorithm is required. The aforementioned works by Bénézit et al. (2010) and Vaidya et al. (2011) do not prove the exponential convergence (but it is necessary to note that the assumptions on the communication in these two works are random and not deterministic).

The only one term of comparison for this algorithm has been very recently found by the author. It is called Primal-Dual Method of Multipliers (PDMM) Sherson, Heusdens, and Kleijn (2017), it is really recent and has an approach similar to the ADMM. This algorithm can work in an asynchronous and lossy communication scenario. Simulations shows that if the graph is sparse, PDMM can be very competitive, while if the graph is well-connected the algorithm introduced in Chapter 5.9 is faster. However, the convergence rate of the PDMM really depends on the choice of a parameter, and a wrong choice of the latter can really slow down the entire algorithm. Conversely, the algorithm proposed

in this chapter does not require any parameters' tuning. In a multi-agent set-up, where the communication graph might be unknown, this latter aspect is important, as will be shown in the simulation section.

5.2 Problem formulation

Consider a group of N agents which can communicate according to the communication graph $\mathcal{G}$. Until Section 5.9, the communication graph $\mathcal{G}$ is assumed *undirected* and connected. From Section 5.9 instead, the graph is assumed *directed* and strongly connected.

Remark 5.2.1. In some distributed systems, such as Wireless Sensor Networks, the communication graph is often undirected, in the sense that a node can transmit to any node from which it can receive. However, communication is typically only half-duplex, i.e., two nodes cannot communicate simultaneously, so that protocols with multiple communication rounds and reliable acknowledge (ACK) mechanisms are needed for bidirectional communication. This, in turn, requires pairwise synchronization and results in substantial delays; as so, dealing with an undirected graph as a directed one can be valuable.

In the following, first the average consensus problem, and then the minimization of the sum of quadratic cost functions problem are presented. Their relationship is then explored. To ease the exposition, it is assumed that for the consensus problem the quantities to be averaged are scalars, while for the quadratic problem, the local cost functions are assumed to be scalar. It is possible to easily extend the work to the multidimensional case, as briefly shown in Section 5.8.

Average consensus problem

Assume that each node $i \in \{1, \dots, N\}$ has a private scalar quantity $v_i \in \mathbb{R}$, which can be collected in vector $\boldsymbol{v} \in \mathbb{R}^N$. The average consensus problem corresponds to the distributed evaluation of the mean of these v_i, that is the evaluation of

$$\bar{v} = \frac{\sum_i v_i}{N} = \frac{\mathbf{1}^\top \boldsymbol{v}}{N}.$$

Each node has to evaluate $\bar{v}$ only exchanging information between its neighbors according to the graph $\mathcal{G}$.

Minimization of the sum of quadratic cost functions

In this quadratic problem, each agent is endowed with a private scalar quadratic cost function

$$f_i \colon \mathbb{R} \to \mathbb{R}, \qquad f_i(x) = \frac{1}{2} a_i (x - \theta_i)^2, \tag{5.1}$$

where $a_i > 0$, $\theta_i \in \mathbb{R}$, $i = 1, \dots, N$. The $N \times 1$ vectors $\boldsymbol{\theta}$ and $\boldsymbol{a}$ (thereafter called *curvature vector*) collect the values θ_i and a_i, respectively. Considering now the global cost function $f \colon \mathbb{R} \to \mathbb{R}$, which is the sum of the cost functions (5.1) of each agent,

$$f(x) = \sum_{i=1}^{N} f_i(x),$$

each agent's aim is to solve the following problem

$$x^* = \arg\min_{x \in \mathbb{R}} f(x). \tag{5.2}$$

The minimizer $x^* \in \mathbb{R}$ of $f(x)$ has to be evaluated by the agents in a distributed way. Namely, each agent can only communicate with its respective neighbors defined by $\mathcal{G}$.

Relationship between the two problems

The minimizer of Problem (5.2) has the following closed form

$$x^* = \frac{\sum_{i=1}^{N} a_i \theta_i}{\sum_{i=1}^{N} a_i}. \tag{5.3}$$

This quantity can clearly be obtained as the ratio between the following two quantities

$$\frac{\sum_{i=1}^{N} a_i \theta_i}{N}, \qquad \frac{\sum_{i=1}^{N} a_i}{N}.$$

These latter quantities are the averages of $a_1\theta_1, \dots, a_N\theta_N$ and of $a_1, \dots, a_N$, respectively. As a consequence, the minimizer of Problem (5.2) can be recovered solving two average consensus problems.

On the other hand, if in Problem (5.2) $a_i = 1$ for all $i \{1, \dots, N\}$, then the minimizer of the problem is

$$x^* = \frac{\sum_{i=1}^{N} \theta_i}{N}$$

which corresponds to the average consensus of the quantities $\theta_1, \dots, \theta_N$.

These relationships justify saying that being able to solve the average consensus

problem implies being able to solve the minimization of the sum of quadratic cost functions, and vice-versa.

5.3 Consensus based algorithms: standard consensus (C)

The algorithms introduced in this section and in the following one aim at solving the consensus problem. If one wants to solve a given quadratic problem of the form (5.2), it is necessary to solve two consensus algorithms in parallel, one to find the average of $\boldsymbol{a} \odot \boldsymbol{\theta}$, and one to find the average of $\boldsymbol{a}$. Then the minimizer can be found computing the ratio between these two quantities.

Formally, denote by $x_i(k)$ the estimate of the mean $\bar{v}$ stored in memory by node i at time k, and define the vector $\boldsymbol{x}(k) := [x_1(k), \dots, x_N(k)]^T \in \mathbb{R}^N$. To solve the consensus problem means to develop an algorithm such that

$$\lim_{k\to\infty} x_i(k) = \bar{v}, \ i \in \{1, \dots, N\} \quad \Leftrightarrow \quad \lim_{k\to\infty} \boldsymbol{x}(k) = \bar{v}\mathbf{1}_N$$

and such that the update of $x_i(k)$ depends only on quantities that belong to the neighbors of node i in $\mathcal{N}_i$.

A well known algorithm to compute the mean of a vector in a distributed way is the average consensus algorithm Garin and Schenato (2010); Olshevsky and Tsitsiklis (2009). Given the communication graph $\mathcal{G}$, construct a stochastic matrix P consistent with the graph (see the part on graph notation in Section A). Since $\mathcal{G}$ is undirected and connected, this P can be built symmetric and primitive (and so its eigenvalues λ_i are real and such that $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_N > -1$). Due to the Perron-Frobenius theorem (Horn and Johnson, 1985, Ch. 8), it holds

$$\lim_{t\to\infty} P^t = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top. \tag{5.4}$$

To obtain in a distributed way the mean of the elements of a vector $\boldsymbol{v} \in \mathbb{R}^N$, it is enough to apply the following iterative scheme

$$\begin{cases} \boldsymbol{x}(k+1) = P\boldsymbol{x}(k) \\ \boldsymbol{x}(0) = \boldsymbol{v} \end{cases}, \quad k \geq 0.$$

As a matter of fact, introducing the quantity $\boldsymbol{x}^* = \bar{v}\mathbf{1}_N$, the Expression (5.4) implies that

$$\lim_{t\to\infty} \boldsymbol{x}(k) = \mathbf{1}_N \frac{\mathbf{1}_N^\top \boldsymbol{v}}{N} = \boldsymbol{x}^*$$

The convergence rate of the consensus algorithm, denoted as ρ_C, is determined by the essential spectral radius (ESR) of matrix P, see Olshevsky and Tsitsiklis (2009); Carli, Fagnani, Speranzon, and Zampieri (2008). In particular, for a positive constant c depending only on $\boldsymbol{x}(0)$, it holds

$$\|\boldsymbol{x}^* - \boldsymbol{x}(k)\|_2 \leq c\rho_C^k, \quad \forall t \geq 0.$$

In the simulation section, matrices P are constructed via the Metropolis-Hastings weights (MHW), Boyd et al. (2004), since they are easy to compute and above all can be calculated locally (each agent needs only to know the number of its neighbors and their degree). Moreover, compared to the Laplacian weights selection (another simple and popular way to build P), it has in general better convergence rate Garin and Schenato (2010); Boyd et al. (2004). Matrix P can be built to have minimal essential radius Boyd et al. (2004), however to do so it is necessary to solve an optimization problem which needs global information on the system (and so has to be solved in a centralized manner). Usually, with MHW, dense graphs far from being bipartite (e.g. random geometric graphs with high distance threshold as in Section 5.7, or also graphs with many randomly selected edges Boyd et al. (2004)) have ρ_C close to 0 (and exactly 0 if $\mathcal{G}$ is complete), while for sparse graphs (i.e. graphs with a small number of edges) ρ_C tends to 1. A graph, whose corresponding doubly-stochastic matrix has a small ESR, is called *well-connected*.

5.4 Consensus based algorithms: accelerated consensus (AC)

Standard consensus is an easy algorithm to solve the average problem, but its performance can be poor, especially when the graph is very sparse.

To improve the convergence rate (while keeping the simplicity of the algorithm), the authors in Muthukrishnan et al. (1998) introduced the use of memory. Other papers resorting to this idea are Oreshkin et al. (2010); Liu and Morse (2011). Given a matrix P consistent with $\mathcal{G}$, the accelerated consensus algorithm to evaluate $\bar{m}$ has the following scheme Muthukrishnan et al. (1998); Liu and Morse (2011).

$$\begin{cases} \boldsymbol{x}(k+1) = \beta P\boldsymbol{x}(k) + (1-\beta)\boldsymbol{x}(k-1) \\ \boldsymbol{x}(0) = \boldsymbol{x}(-1) = \boldsymbol{v} \end{cases}, \quad k \geq 0. \tag{5.5}$$

The scalar β has to be selected inside the interval $(0, 2)$ to have a converging algorithm.

Introducing the augmented state $\boldsymbol{z}(k) = [\boldsymbol{x}(k)^\top \ \boldsymbol{x}(k-1)^\top]^\top$, the dynamic of (5.5) can be rewritten as

$$\boldsymbol{z}(k+1) = \begin{bmatrix} \beta P & (1-\beta)I_N \\ I_N & \mathbf{0} \end{bmatrix} \boldsymbol{z}(k) := Q\boldsymbol{z}(k), \tag{5.6}$$

with initial condition $\boldsymbol{z}(0) = [\boldsymbol{v}^\top \ \boldsymbol{v}^\top]^\top$. Note that matrix Q has an eigenvalue 1 with corresponding eigenvector $\mathbf{1}_{2N}$, and selecting $0 < \beta < 2$ this eigenvalue is the biggest in absolute value.

The aim is to select the parameter β in order to minimize the convergence rate of the algorithm, which corresponds to minimize the ESR of Q. The following result holds Muthukrishnan et al. (1998); Liu and Morse (2011):

Proposition 5.4.1. *Given ρ_C, the convergence rate of the consensus algorithm ruled by matrix P, the optimal convergence rate of the accelerated consensus, denoted by ρ_{AC}, is achieved setting β equal to*

$$\beta^* := \frac{2}{1+\sqrt{1-\rho_C^2}} > 1,$$

and it is equal to

$$\rho_{AC} := \frac{\rho_C}{1+\sqrt{1-\rho_C^2}} \leq \rho_C < 1, \tag{5.7}$$

where $\rho_{AC} = \rho_C$ if and only if $\rho_C = 0$.

Since $\rho_{AC} < 1$ and $\mathbf{1}_{2N}$ is the eigenvector related to eigenvalue 1 of Q, it holds that $\lim_{k\to\infty} \boldsymbol{x}(k) = \hat{\alpha}\mathbf{1}_N$, $\hat{\alpha} \in \mathbb{R}$. Due to the fact that the update (5.5) is such that the average of the elements of $\boldsymbol{x}(k)$ is equal to $\bar{v}$, $\forall t > 0$, then $\hat{\alpha} = \bar{v}$.

Note that the evaluation of β^* requires the knowledge of ρ_C, which can be obtained in a decentralized way Oreshkin et al. (2010).

Remark 5.4.2. The update rule of this accelerated consensus has similarities with a proportional and derivative feedback control. Moreover, a strong similarity can be found with the Heavy-Ball method Ghadimi, Shames, and Johansson (2012).

5.5 Lagrangian based algorithms: dual ascent method (DA)

To solve (5.2) in a distributed way, it is possible to recast the problem by introducing suitable equality constraints. The Lagrangian function which includes these constraints is then constructed, in order to solve the problem. The first Lagrangian-based algorithm presented is the dual ascent approach (Boyd et al., 2011, Ch 2). In the following section

ADMM Boyd et al. (2011) will be analyzed. The analytical results presented in these two sections are obtained upon restricting (in a significant way) the cost functions. Namely, the curvature vector satisfy $\boldsymbol{a} = \bar{a}\mathbf{1}_N$. The general case $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$ is very difficult to be treated analytically, and will be analyzed in Section 5.7 through simulations.

Assume each agent stores in memory a copy of the optimization variable x denoted as x_i. Let $\boldsymbol{x} = [x_1, \dots, x_N]^T$. Then, according to the dual ascent approach (Boyd et al., 2011, Ch 2), the solution of problem (5.2) is equivalent to the constrained problem

$$\begin{aligned} &\arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \frac{1}{2}\sum_{i=1}^{N} a_i\,(x_i-\theta_i)^2 \\ &\text{subject to } x_i = x_j, \quad \forall j \in \mathcal{N}_i,\, \forall i \in \{1,\dots,N\} \end{aligned} \quad . \tag{5.8}$$

Let $\boldsymbol{x}^* = [x_1^*, \dots, x_N^*]$ be the optimal solution of Problem (5.8). Then, the constraints in (5.8) ensure that the solution $\boldsymbol{x}^*$ has the form $x^*\mathbf{1}_N$. The Lagrangian for this problem is

$$\mathcal{L}(\boldsymbol{x}, \Lambda) = \frac{1}{2}\sum_{i=1}^{N} a_i(x_i-\theta_i)^2 + \sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i} \lambda_{ij}(x_i - x_j), \tag{5.9}$$

where $\Lambda \in \mathbb{R}^{N\times N}$ collects the different Lagrangian multipliers λ_{ij}, which are non-zero if and only if $j \in \mathcal{N}_i$.

The dual ascent method is an iterative algorithm that alternates between a maximization of the Lagrangian with respect to Λ, keeping fixed $\boldsymbol{x}$, and a minimization step on the Lagrangian with respect to $\boldsymbol{x}$, keeping fixed Λ. Its functioning is strictly related to dual theory in optimization (see Boyd and Vandenberghe (2004)).The corresponding maximizer and minimizer have, for the particular Lagrangian in Equation (5.9), a closed form. Namely, the update steps for $\lambda_{ij}(k)$ and for $x_i(k)$ are the following

$$\begin{cases} \lambda_{ij}(k+1) = \lambda_{ij}(k) + \epsilon[x_i(k) - x_j(k)], & (5.10) \\ x_i(k+1) = \theta_i - \dfrac{\sum_{j\in\mathcal{N}_i}[\lambda_{ij}(k+1) - \lambda_{ji}(k+1)]}{a_i}, & (5.11) \end{cases}$$

where ϵ is a (sufficiently small) fixed step size and $\boldsymbol{x}(0)$ and $\lambda_{ij}(0)$ are given initial conditions.

Recalling that matrix $A_{\mathcal{G}}$ is the adjacency matrix of graph $\mathcal{G}$, it is possible to introduce the matrices $A_{\mathcal{G}}^- = A_{\mathcal{G}} - I_N$ and $L_{\mathcal{G}} = \text{diag}(A_{\mathcal{G}}^-\mathbf{1}_N) - A_{\mathcal{G}}^-$, where the latter corresponds to the Laplacian of the graph $\mathcal{G}$ (without the self-loops). The real eigenvalues γ_i of $L_{\mathcal{G}}$ satisfy $\gamma_1 = 0 \leq \gamma_2 \leq \cdots \leq \gamma_N$. If all $x_i(0)$ and $\lambda_{ij}(0)$ are chosen equal to 0, then the

dynamics of the variables $x_i(k)$ given by (5.10) and (5.11) can be written in compact form as $\boldsymbol{x}(k+1) = V\boldsymbol{x}(k)$, $\boldsymbol{x}(1) = \theta$ where matrix V is given by:

$$V := I_N - 2\epsilon \operatorname{diag}(\boldsymbol{a})^{-1} L_{\mathcal{G}}, \tag{5.12}$$

Matrix V has an eigenvalue in 1 with eigenvector $\mathbf{1}_N$, and if ϵ is such that all the other eigenvalues are in modulus smaller than 1, then the algorithm converges.

Remark 5.5.1. If $\epsilon < \epsilon_{\max}$, with $\epsilon_{\max} := \max_i \left\{ \frac{a_i}{2|\mathcal{N}_i|} \right\}$, the dual ascent is a consensus algorithm. Although the choice $\epsilon < \epsilon_{\max}$ guarantees convergence, the best choice in terms of convergence rate could be achieved for a value $\epsilon^* \geq \epsilon_{\max}$.

If the curvature vector is $\boldsymbol{a} = \bar{a}\mathbf{1}_N$, it is possible to evaluate the optimal value for the step size ϵ. The proofs of this and the following proposition can be found in Appendix D.1.

Proposition 5.5.2. *Let $\gamma_1 = 0 \leq \gamma_2 \leq \cdots \leq \gamma_N$ be the eigenvalues of $L_{\mathcal{G}}$ and let $\boldsymbol{a}$ be equal to $\bar{a}\mathbf{1}_N$. Then the optimal value for ϵ and the corresponding convergence rate for dual ascent are*

$$\epsilon^* = \frac{\bar{a}}{\gamma_2 + \gamma_N}, \qquad \rho_{DA} = \frac{\gamma_2 - \gamma_N}{\gamma_2 + \gamma_N} < 1.$$

In case $\mathcal{G}$ is a $d-$regular graph and the curvature vector is $\boldsymbol{a} = \bar{a}\mathbf{1}_N$, it is possible to compare the consensus and the dual ascent algorithms. In fact, in this case the matrix P (built using the MHW) for the consensus algorithm is $P = \frac{1}{d+1} A_{\mathcal{G}}$. Starting from (5.12), V can be rewritten as

$$V = \left(1 - \frac{2(d+1)\epsilon}{\bar{a}}\right) I_N + \frac{2(d+1)\epsilon}{\bar{a}} P, \tag{5.13}$$

and the following proposition holds:

Proposition 5.5.3. *Let $\mathcal{G}$ be a $d-$regular graph, $\boldsymbol{a}$ is equal to $\bar{a}\mathbf{1}_N$. Let $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_N > -1$ be the eigenvalues of matrix P (consistent with $\mathcal{G}$) and let ρ_C be the ESR of matrix P. Then, the optimal convergence rate ρ_{DA} for the dual ascent algorithm is*

$$\rho_{DA} = \frac{\lambda_2 - \lambda_N}{2 - \lambda_2 - \lambda_N} \leq \rho_C. \tag{5.14}$$

Moreover, defining $\xi = \frac{d-1}{d+1}$, the following bounds hold

$$\rho_{DA} \geq \frac{\rho_C}{2 + \rho_C}, \qquad \text{if } \ 0 \leq \rho_C \leq \xi, \tag{5.15}$$

$$\rho_{DA} \geq \frac{\rho_C}{2 - \rho_C}, \qquad \text{if } \ \xi < \rho_C \leq 1. \tag{5.16}$$

5.6 Lagrangian based algorithms: Alternating Direction Method of Multipliers (ADMM)

The *Alternating Direction Method of Multipliers* (ADMM) is a well known and heavily employed algorithm to solve many different optimization problems (see the survey Boyd et al. (2011) and reference therein). To solve (5.2) using ADMM, as done for the dual ascent each agent is allowed to store in memory a copy of the optimization variable x denoted as x_i, but, in this case, also an auxiliary vector $\boldsymbol{z} \in \mathbb{R}^N$ is introduced, and the problem becomes

$$\begin{cases} \underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to } x_i = z_j \quad \forall j \in \mathcal{N}_i^+, \forall i \in \{1,\dots,N\} \end{cases}. \tag{5.17}$$

The constraints in this reformulation assure that the optimal solution $\boldsymbol{x}^*$ is such that $\boldsymbol{x}^* = x^* \mathbf{1}_N$. Differently from the Lagrangian employed for the dual ascent method (Equation (5.9)) ADMM uses an augmented Lagrangian with Lagrangian multipliers $\lambda_{ij} \in \mathbb{R}$ and penalty parameters $w_{ij} > 0$, $i,j = 1,\dots,N$, $(i,j) \in \mathcal{E}$,

$$\mathcal{L}_W(\boldsymbol{x},\boldsymbol{z},\Lambda) = \sum_{i=1}^{N} \frac{1}{2} a_i (x_i - \theta_i)^2 + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i^+} \lambda_{ij}(x_i - z_j) + \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i^+} w_{ij}(x_i - z_j)^2. \tag{5.18}$$

$W \in \mathbb{R}^{N \times N}$ and $\Lambda \in \mathbb{R}^{N \times N}$ contain respectively the w_{ij} and the λ_{ij} (the w_{ij} and λ_{ij} corresponding to couples (i,j) not belonging to $\mathcal{E}$ are equal to 0). This formulation for the augmented Lagrangian is a bit different from the standard ADMM, where $w_{ij} = \bar{w}$ for all $i,j = 1,\dots,N$. The choice done here follows Erseghe et al. (2011).

Similarly to the dual ascent, the ADMM alternates between minimization and maximization steps on the augmented Lagrangian. In particular, an iteration is as follows:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \arg\min_{\boldsymbol{x}} \mathcal{L}_W(\boldsymbol{x}, \boldsymbol{z}(k), \Lambda(k)), \\ \boldsymbol{z}(k+1) &= \arg\min_{\boldsymbol{z}} \mathcal{L}_W(\boldsymbol{x}(k+1), \boldsymbol{z}, \Lambda(k)), \\ \Lambda(k+1) &= \arg\min_{\Lambda} \mathcal{L}_W(\boldsymbol{x}(k+1), \boldsymbol{z}(k+1), \Lambda). \end{aligned}$$

Again, due to the particular form of the augmented Lagrangian (5.18), these update can

be rewritten in closed form:

$$\begin{cases} x_i(t+1) = \dfrac{a_i\theta_i + \sum_{j\in\mathcal{N}_i^+}\left[w_{ij}z_j(k) - \lambda_{ij}(k)\right]}{a_i + \sum_{j\in\mathcal{N}_i^+} w_{ij}}, & (5.19) \\ z_j(t+1) = \dfrac{\sum_{i\in\mathcal{N}_j^+} w_{ij}x_i(k+1) + \sum_{i\in\mathcal{N}_j^+}\lambda_{ij}(k)}{\sum_{i\in\mathcal{N}_j^+} w_{ij}}, & (5.20) \\ \lambda_{ij}(k+1) = \lambda_{ij}(k) + w_{ij}\left(x_i(k+1) - z_j(k+1)\right). & (5.21) \end{cases}$$

ADMM converges for any choice of the initial conditions for $x_i(0), z_i(0)$ and $\lambda_{ij}(0)$. However, choosing suitable initial conditions (namely $z_i(0) = 0$ and $\lambda_{ij}(0) = 0$ for all $i, j \in \{1, \ldots, N\}$), this update scheme can be rewritten in a matrix form which only involves the variable $\boldsymbol{x}$ at time k and $k-1$ (see Appendix D.2 for the detailed calculations). Defining matrices

$$D := \operatorname{diag}(W\mathbf{1}_N)\operatorname{diag}(\boldsymbol{a} + W\mathbf{1}_N)^{-1},$$

$$U := \operatorname{diag}(\boldsymbol{a} + W\mathbf{1}_N)^{-1}W\operatorname{diag}(\mathbf{1}_N^\top W)^{-1}W^\top - D,$$

$$M := I_N + D + 2U, \qquad K := D + U,$$

the matrix form of the ADMM is

$$\begin{cases} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k) \end{bmatrix} = \underbrace{\begin{bmatrix} M & -K \\ I_N & \mathbf{0}_N \end{bmatrix}}_{F} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{x}(k-1) \end{bmatrix}, \quad t \geq 1. \\ \boldsymbol{x}(0) = 0, \quad \boldsymbol{x}(1) = (I_N - D)\theta \end{cases} \tag{5.22}$$

Matrix F determines the rate of convergence of ADMM through its ESR.

In the following analytical study, matrix W is selected as $W = \mu P$, with $\mu > 0$ and P a symmetric stochastic matrix consistent with the graph $\mathcal{G}$. The ADMM obtained with this choice for W is denoted as ADMM$_P$, and its convergence rate as ρ_{ADMM_P}. This particular choice is done to carry out a comparison with the consensus algorithm (as will be soon clear).

In truth, it is possible to select W in different ways. The choice, $W = \varphi A_{\mathcal{G}}$, $\varphi > 0$, according to which all the penalty parameters w_{ij} are equal, is the one usually done in the literature. In this particular context, however, it does not allow for an immediate comparison with the standard consensus. In Section 5.7, some simulations will also involve ADMM with this choice for matrix W. To avoid confusion, this (standard) version of ADMM is denoted as ADMM$_A$, and its convergence rate as ρ_{ADMM_A}. The simulations will show that the behavior of ADMM$_A$ is similar to the one of ADMM$_P$.

Remark 5.6.1. Note that, for $d-$regular graphs, constructing matrix P using the MHW implies that ADMM$_P$ is equivalent to ADMM$_A$.

In case $\boldsymbol{a} = \bar{a}\mathbf{1}_N$, that is in case all the quadratic cost functions have the same curvature, it is possible to analytically determine the convergence rate of the ADMM$_P$. The matrices K and M obtained in this scenario are

$$K = \frac{\mu}{\bar{a}+\mu}P^2, \quad M = \left(1 - \frac{\mu}{\bar{a}+\mu}\right)I_N + 2\frac{\mu}{\bar{a}+\mu}P^2. \tag{5.23}$$

Adopting the quantity $\delta = \frac{\mu}{\bar{a}+\mu}$, $0 < \delta < 1$, next proposition holds, whose proof is given in Appendix D.3.

Proposition 5.6.2. *If ρ_C is the essential spectral radius of matrix P in (5.23) and if the curvature vector $\boldsymbol{a}$ is equal to $\bar{a}\mathbf{1}_N$, the optimal convergence rate ρ_{ADMM_P} for ADMM$_P$ is achieved using the following value for δ*

$$\begin{cases} \delta^* = \frac{1}{2} & \text{if } \rho_C \leq \frac{1}{\sqrt{2}}, \\ \delta^* = \frac{1}{1+2\sqrt{\rho_C^2-\rho_C^4}} & \text{if } \rho_C > \frac{1}{\sqrt{2}}, \end{cases}$$

which leads to

$$\begin{cases} \rho_{ADMM_P} = \frac{1}{2} & \text{if } \rho_C \leq \frac{1}{\sqrt{2}}, \\ \rho_{ADMM_P} = \frac{\rho_C}{\rho_C+\sqrt{1-\rho_C^2}} & \text{if } \rho_C > \frac{1}{\sqrt{2}}. \end{cases} \tag{5.24}$$

5.7 Analytic and simulative comparison: scalar case

In this section the previous four algorithms are compared with respect to the convergence rate. The first comparison is for the simplified case in which all the curvature of the $f_i(x)$ are equal, i.e. $\boldsymbol{a} = \bar{a}\mathbf{1}_N$. In this scenario the comparison is analytic. Then, the more general case $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$ is examined through simulations.

Case: $\boldsymbol{a} = \bar{a}\mathbf{1}_N$

Under the equal curvature scenario, by combining the results in the previous section, the following proposition holds:

Proposition 5.7.1. *When the curvature vector $\boldsymbol{a}$ is equal to $\bar{a}\mathbf{1}_N$, accelerated consensus has the best performance with respect to the consensus algorithm and ADMM.*

An asymptotic result on ρ_{ADMM_P} is now provided. Consider a sequence of undirected connected graphs $\mathcal{G}_N$ of increasing size N, and, for each $\mathcal{G}_N$, let P_N be the stochastic matrix consistent with the graph. Assume the following property.

Assumption 5.7.2. Consider the sequence of matrices P_N associated to the sequence of graphs $\mathcal{G}_N$ above described, and assume that $\rho_{C_N} = 1 - \varepsilon(N) + o(\varepsilon(N))$ with $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that $\varepsilon(N) \to 0$ as $N \to \infty$.

Important families of matrices satisfying Assumption 5.7.2 are those built over the d-dimensional tori and the Cayley graphs (see Carli et al. (2008)). These graphs exhibit important spectral similarities with the random geometric graphs Boyd, Ghosh, Prabhakar, and Shah (2006), which is a family of graphs that has been successfully used to model wireless communication in many applications Franceschetti and Meester (2008). The following result gives the asymptotic behavior of $\rho_{ADMM_{P\,N}}$ and ρ_{AC_N} (the subscript N denotes the dependence on N).

Proposition 5.7.3. *Under Assumption 5.7.2 the convergence rate of $ADMM_P$ and of the accelerated consensus asymptotically assume the same value, that is*

$$\lim_{N\to\infty} \rho_{ADMM_{P\,N}} = \lim_{N\to\infty} \rho_{AC_N} = 1 - \sqrt{2\varepsilon(N)}.$$

Proof. When $\varepsilon(N)$ is sufficiently small, the following calculations hold

$$\begin{aligned}\rho_{ADMM_{P\,N}} &= \frac{\rho_{C_N}}{\rho_{C_N} + \sqrt{1-\rho_{C_N}^2}} \simeq \frac{1-\varepsilon(N)}{1-\varepsilon(N)+\sqrt{2\varepsilon(N)}} \\ &\simeq \frac{(1-\varepsilon(N))(1-\sqrt{2\varepsilon(N)}}{1-2\varepsilon(N)} \simeq 1-\sqrt{2\varepsilon(N)} + o(\sqrt{\varepsilon(N)}).\end{aligned}$$

Similar calculations hold for ρ_{AC_N}. ∎

Figure 5.1 shows the convergence rate of standard consensus, accelerated consensus (Proposition 5.4.1), ADMM$_P$ (Proposition 5.6.2) and the lower bound for the convergence rate of dual ascent (Proposition 5.5.3) for the case of 4-regular graphs, all built starting from the same matrix P. The figure also provides the upper bounds given in Shi et al. (2014); Makhdoumi and Ozdaglar (2016) for the ADMM, bounds which have been obtained again for a 4-regular graph[1].

The figure allows to easily verify the statement of Propositions 5.7.1 and 5.7.3.

The bounds provided in Shi et al. (2014); Makhdoumi and Ozdaglar (2016) have been applied to the specific problem of this chapter in case of regular graphs (since

[1]The bounds in Shi et al. (2014); Makhdoumi and Ozdaglar (2016) depend only on λ_2, the second eigenvalue of the stochastic matrix P consistent with the communication graph and built using the MHW. So, to evaluate the bounds, in the abscissa of the figure one uses ρ_C if $\rho_C = \lambda_2$; if ρ_C is instead determined by λ_N, to determine the value of the bounds one has to use as abscissa the value of λ_2 (in a $d-$regular graph, λ_2 is always positive since $\lambda_2 \geq 1/(d+1)$).

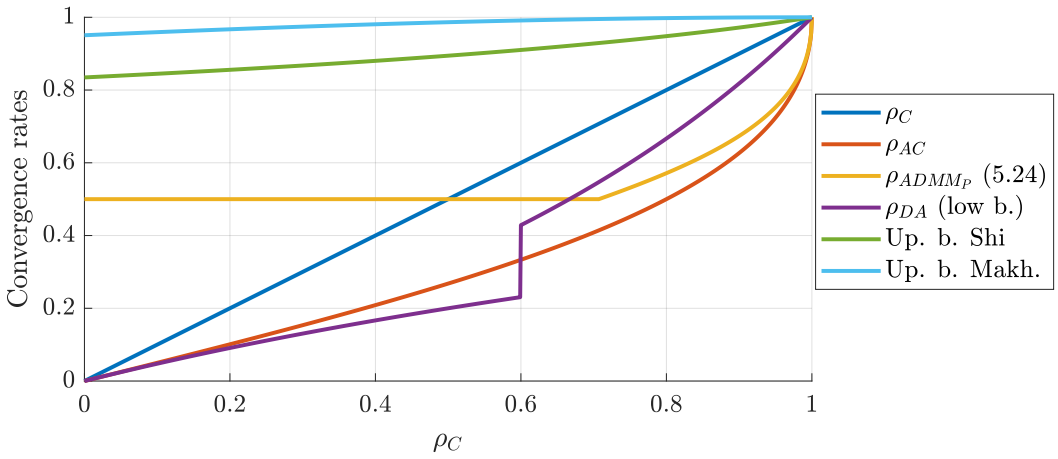


Figure 5.1: Comparison between consensus, AC, DA and ADMM$_P$ when $\boldsymbol{a} = \bar{a}\mathbf{1}_N$ according to the analytical results found in Propositions 5.4.1, 5.5.3 and 5.6.2. The lower bound shown for DA is for a $d-$regular graph with $d = 4$ (note that in this case $\xi = 0.6$); for bigger value of d, ξ gets nearer to one. The figure also reports the bounds for ADMM given in Shi et al. (2014); Makhdoumi and Ozdaglar (2016), evaluated again in a d-regular graph ($d = 4$).

this allows to express the bounds as functions of ρ_C). In these papers ADMM is used to solve problem (5.2) where f is the sum of generic convex functions f_i (each work proposes a version of ADMM which is slightly different from the one proposed here). The bounds obtained are very conservative in the quadratic scenario. Conversely, here the convergence rate of the ADMM$_P$ is given, even if for a very restrictive scenario. The convergence rate shows that the ADMM$_P$ "saturates" when the underlying graph is connected enough, as opposed to what happens to the other algorithms analyzed. This interesting aspect, which can guide in the selection of the algorithm to use, is not evident at all in the bounds of Shi et al. (2014); Makhdoumi and Ozdaglar (2016), since they are only upper bounds. Note that also in Teixeira et al. (2013) the convergence rate of standard ADMM does not decrease as the connectivity of the graph increases; in the same paper however it is shown that the use of memory and of some proper scaling in the matrix of the constraints can result in an improvement of the convergence rate, which decreases as the connectivity increases. However these modifications of ADMM imply the presence of additional parameters to be tuned and are not considered here.

Figure 5.2 contains some simulations to better study the behavior of the DA algorithm with respect to the analytical lower bound found in Proposition 5.5.3. The figure presents the convergence rate for the DA for different kinds of communication graphs, namely random regular graphs, random geometric graphs and bipartite graphs. The number of agents is the same for all the simulations. These simulations shows that the bound in

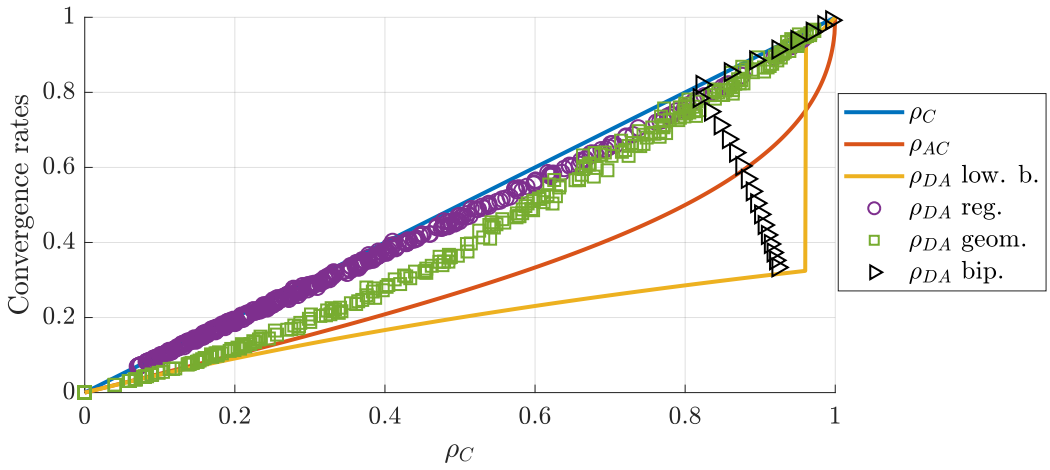


Figure 5.2: Comparison between the convergence rate of consensus, AC and DA for a system with $N = 50$ nodes, when $\boldsymbol{a} = \bar{a}\mathbf{1}_{50}$. The parameter ϵ for the DA is selected to obtain fastest convergence using a line search approach. The lower bound shown for DA is for a 25−regular graph. Reg. stands for regular, and the simulations involve d-regular random graphs with increasing d (from the upper right to the lower left). For each choice of d, 30 different realizations of the graph are obtained. Geom. stands for geometric, and for these simulations the distance threshold changes from 0.3 to 1.2 (again from the upper right to the lower left). For each choice of the distance threshold 30 different realizations of the graph are obtained. Finally, bip. stands for bipartite; in this case the graph is chosen d−regular, and from the upper right to the lower right d is increased (the lowest point represents the complete bipartite graph).

(5.15) is tight and is achieved for bipartite complete graph as N grows. As a consequence, DA can have a performance that is better than AC. However, when the regular graph is randomly chosen, its convergence rate is far from the lower bound. Therefore, for a random d−regular graph, the best strategy to solve the problem is likely to be the accelerated consensus. The simulations on random geometric graphs again show that DA has a better convergence rate with respect to the consensus algorithm, but not with respect to the accelerated consensus. It is also interesting to note that the convergence rate for the consensus algorithm in case of a complete bipartite graph is definitely slow, even though the number of edges is consistent.

As shown in the following subsection, applying the algorithms when the curvature vector is not homogeneous, the ADMM maintains the "saturating" behavior, while the performance of the DA terribly decreases.

Case: $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$

To study the performance of the algorithms for more generic quadratic cost functions (namely, for cost functions with different curvatures) some simulations have been carried

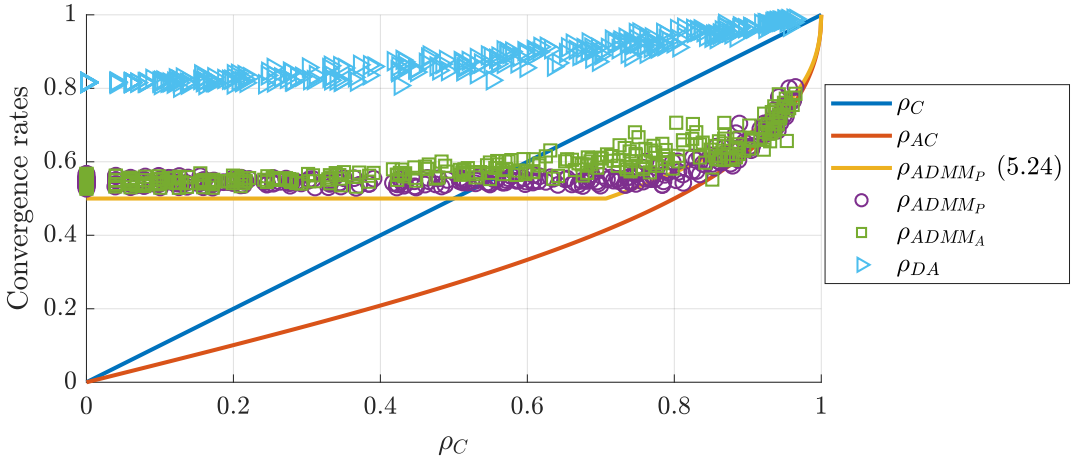


Figure 5.3: Convergence rate of AC, DA and ADMM as a function of the convergence rate of consensus with $\boldsymbol{a} \neq \bar{a}1_N$, $a_{\min} = 1$ and $a_{\max} = 10$. The graphs are random geometric with $N = 50$ nodes and a varying distance threshold. For each value of the distance threshold 30 different realizations of the graph have been made. From the top right to the bottom left, the distance threshold increases (and so the graph becomes more and more connected). The figure contains also the analytic convergence rate for $\boldsymbol{a} = \bar{a}\mathbf{1}_N$. When the distance threshold for the random geometric graph is 0.8 or greater, generally the ESR of the corresponding matrix P (built using MHW) is smaller than 1/2.

out, presented in Figure 5.3.

In the simulations, the curvature vector $\boldsymbol{a}$ is generated randomly. Given $a_{\min}$, $a_{\max} \in \mathbb{R}$ the minimum and maximum values of $\boldsymbol{a}$, its other elements are randomly selected in the interval $a_{\min}$ and $a_{\max}$ (the order is also randomly chosen).

The simulations consider random geometric graphs, due to their usefulness in applications. Moreover, carrying out simulations using other types of randomly generated graphs, no significant differences from the results of Figure 5.3 can be obtained.

In this scenario for the curvature vector, to select the optimal values of ϵ, μ_P and μ_A, which minimize the convergence rate of DA, ADMM$_P$ and ADMM$_A$, respectively, a line search approach is employed.

The bounds for ADMM given by Shi et al. (2014); Makhdoumi and Ozdaglar (2016), are even more conservative than the ones in Figure 5.1 and so have not been reported.

Analyzing the figure, it is easy to note that DA applied to scenarios where $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$ has a convergence rate which is much worse than the consensus, and its behavior is completely different from the one which is obtained when $\boldsymbol{a} = \bar{a}\mathbf{1}_N$ (see Figure 5.2). Also, repeating the simulations for the bipartite graphs, the same degradation is observed. Therefore, the bound given by Proposition 5.5.3 is not significant when the the condition

$\boldsymbol{a} = \bar{a}\mathbf{1}_N$ is not met.

Concerning ADMM$_P$, instead, Figure 5.3 shows that, when $\boldsymbol{a} \neq \mathbf{1}_N$, the behavior of its convergence rate is qualitatively similar to the one obtained for $\boldsymbol{a} = \mathbf{1}_N$. The same figure contains the convergence rate for the ADMM$_A$. it is easy to verify that ADMM$_A$ has a convergence rate similar to the one of the ADMM$_P$. Thus the results of Section 5.7 are still consistent in case of $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$ for ADMM$_P$, while this is not true for DA.

As general observations, AC has the best performance in almost all the simulations carried out. Moreover, it is possible to affirm that as long as the graph is sparse, ADMM$_P$ is a good algorithm to solve the optimization problem, while for dense graph AC is preferable. As a matter of fact ρ_{ADMM} does not decrease as the graph becomes more and more connected, differently from what is observed for the consensus based algorithms and also for DA when $\boldsymbol{a} = \bar{a}\mathbf{1}_N$.

When $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$, an important difference between the algorithms can be highlighted. Consensus based algorithms are independent from the curvature vector $\boldsymbol{a}$ both for the convergence rate and for the tuning of the possible parameters. For Lagrangian based algorithms instead, the convergence rate and also the optimal tuning of the free parameters depend on the specific value of the curvatures a_i. Therefore the knowledge of the graph is not sufficient to determine alone their convergence rate and the optimal parameters.

Both analyzed cases show that accelerated consensus (with optimized parameter β^*) is (almost always) faster than all the other methods analyzed. However, it is correct to remind that consensus based algorithms need two consensus algorithms running in parallel to estimate the variable $\boldsymbol{x}$. On the other hand, the ADMM algorithm requires a double message exchange per iteration, one before and one after the update (5.19). This need of an additional synchronization can slow down the ADMM iteration when the number of agents increases.

One of the most interesting take home messages is that for dense graph even standard consensus has a better performance than ADMM$_P$. Therefore, ADMM$_P$ is a good algorithm to apply in graphs with a small number of edges, but it becomes unsuitable for graphs which are highly connected and far from being bipartite. Finally, if $\boldsymbol{a} = \bar{a}\mathbf{1}_N$ and the graph is almost bipartite complete, dual ascent can be a very good algorithm.

5.8 Simulative comparison: multidimensional case

Finally, the analysis is extended to the multivariate quadratic case through some simulations. The functions $f_i \colon \mathbb{R}^p \to \mathbb{R}$ are

$$f_i(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\theta}_i)^\top W_i(\boldsymbol{x} - \boldsymbol{\theta}_i)\,,$$

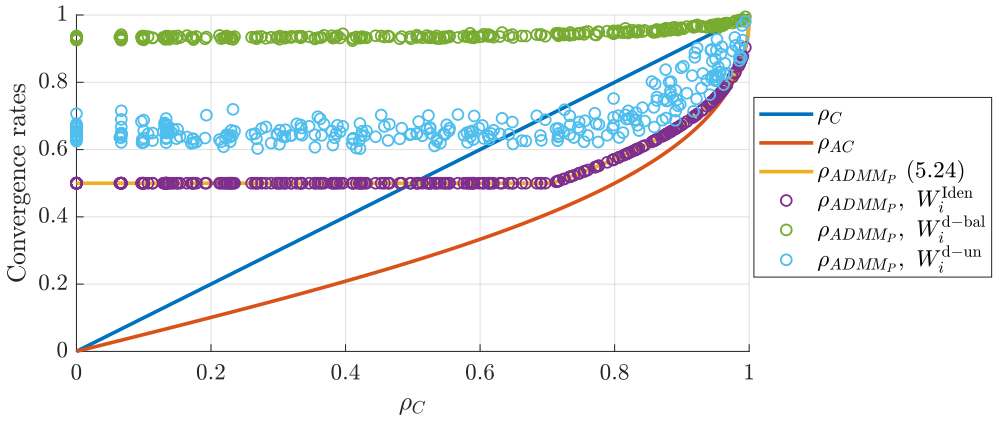


Figure 5.4: Convergence rate of AC and ADMM$_P$ as a function of the convergence rate of consensus for multivariate quadratic cost functions. The curvature matrices influence only the speed of ADMM$_P$. The graphs considered are random geometric with $N = 30$ nodes and a varying distance threshold (from 0.3 to 1.2). For each value of the distance threshold 30 different realization of the graph have been made. From the top right to the bottom left, the distance threshold increases.

with $\boldsymbol{x}, \boldsymbol{\theta}_i \in \mathbb{R}^p$ and $W_i \in \mathbb{R}^{p\times p}$. The global function $f\colon \mathbb{R}^p \to \mathbb{R}$ is the sum of all f_i.

The previous algorithms[2] can be easily and opportunely modified to solve this problem. In particular, the agents running the consensus based algorithms have to evaluate the mean $\bar{W}$ of the matrices $W_1, \dots, W_N$ and the mean $\bar{\boldsymbol{\theta}}_W$ of vectors $W_1\boldsymbol{\theta}_1, \dots, W_N\boldsymbol{\theta}_N$, and the minimizer is evaluated as $\bar{W}^{-1}\bar{\boldsymbol{\theta}}_W$. Each agent have therefore to evaluate the mean of $p^2 + p$ elements.

The optimization parameter μ for ADMM$_P$ that minimizes the convergence rate is again established using a line search.

Figure 5.4 contains the results of the simulations. The communication graphs considered are random geometric graphs, p is set equal to 2 and the curvature matrices used are:

(i) $W_i^{\text{Iden}} = I_2 \in \mathbb{R}^{2\times 2}$;

(ii) $W_i^{\text{d-un}} \in \mathbb{R}^{2\times 2}$ diagonal, with $\left[W_i^{\text{d-un}}\right]_{11}$ equal to $100 \cdot r_1$ and $\left[W_i^{\text{d-un}}\right]_{22}$ equal to r_2, with r_1 and r_2 realizations of a uniform random variable in the interval $[0, 1]$;

(iii) $W_i^{\text{d-bal}} \in \mathbb{R}^{2\times 2}$ obtained from $W_i^{\text{d-un}}$ randomly permuting the diagonal elements.

As for the scalar case, in the consensus based algorithm the curvature matrices play no role in determining the convergence rate of the algorithms, while ADMM$_P$ is strongly

[2]DA is excluded from these simulations since its performance, even for the scalar case when $\boldsymbol{a} \neq \bar{a}\mathbf{1}_N$, is much worse than the performance of the other algorithms.

influenced by them. Simulations show that when the curvature matrices are multiple of the identity matrix the analytical convergence rate of Section 5.6 for $\boldsymbol{a} = \bar{a}\mathbf{1}_N$ is recovered, while for other curvature matrices the behaviors are interesting. In particular if the cost functions are all quite skewed in the same direction, than ADMM$_P$ becomes much slower as compared to itself when the skewness is randomly distributed. The analytical convergence rate for scalar ADMM$_P$ is still indicative when $W_i^{\text{d-bal}}$, while for $W_i^{\text{d-un}}$ the convergence rate is much worse than the analytical result of Proposition 5.6.2. Therefore, in the multivariate case, the choice of the algorithm has to take into account both the graph and the skewness of the agents' cost functions. In all cases accelerated consensus seems to be a better choice in term of convergence rate. However, in the multivariate case AC has to compute means on matrices, requiring therefore to store in memory and to communicate matrices. As a consequence, large values of p can be problematic from a communication and memory requirement perspective.

This multivariate analysis concludes the comparison among the algorithms that can be used to solve the problem examined in this chapter. The algorithms just analyzed assume that the communication is reliable. The following section is devoted to the introduction of an algorithm for the consensus problem able to cope with a non-ideal communication set-up.

5.9 Robust and Asynchronous Average Consensus (ra-AC)

The aim of the algorithm presented in this section is to solve the consensus problem in a more realistic communication scenario. Therefore, the problem's setting is again a group on N agents, each owning a private quantity v_i (collectible in vector $\boldsymbol{v} \in \mathbb{R}^N$) and the aim is to evaluate

$$\bar{v} = \frac{\mathbf{1}_N^\top \boldsymbol{v}}{N} = \frac{\sum_{i=1}^N v_i}{N}. \tag{5.25}$$

The most salient feature of the algorithm is that it has to employ an asynchronous protocol and has to be robust to packet losses. The communication graph $\mathcal{G}$ among the agents meets the following assumption

Assumption 5.9.1. The communication graph $\mathcal{G}$ is time-invariant, **directed** and satisfy the Assumption 2.2.1 on connectivity.

The algorithm designed to solve the consensus problem has to have the following features:

1. *Asymptotic global estimation*: each agent's estimate of the global minimizer has to asymptotically converge to $\bar{v}$.
2. *Peer-to-peer (leaderless)*: each node's update has to consider the limited computational and memory capability available at the node itself and there is no master node among the agents. Moreover, the algorithm can only require communication between one-hop neighbors and it has to assure convergence on any communication graph $\mathcal{G}$ satisfying Assumption 5.9.1.
3. *Distributed*: the update-rule of the local variables at each node has to depend only on the variables stored by the local node and by its neighbors. No multi-hop information exchange is allowed.
4. *Asynchronous*: the algorithm has to allow the agents to perform the update step and the communication step in any moment, without any coordination among the agents.
5. *Lossy broadcast communication without ACK*: the convergence of the algorithm has to be assured even if communication is lossy and broadcast-based. No ACK mechanisms has to be employed.

The *robust and asynchronous Average Consensus* algorithm (ra-AC) takes inspiration from the algorithm presented in Vaidya et al. (2011) but modify it to work in an asynchronous communication set-up. In particular the asynchronous protocol adopted is the broadcast asymmetric one. Namely, only one node and, in a second moment, all its out-neighbors that receive information, update part of their variables at each iteration. In Vaidya et al. (2011), instead, all the nodes at each iteration perform some computations.

As the algorithm in Vaidya et al. (2011), also the ra-AC algorithm is based on the average ratio consensus introduced in Bénézit et al. (2010). Denote again as $x_i \in \mathbb{R}$ the $i-$th agent's estimate of the mean of vector $\boldsymbol{v}$. According to the ratio consensus, the variable x_i is obtained as the ratio of two appropriate scalar quantities $q_i \in \mathbb{R}$ and $s_i \in \mathbb{R}$; the update of q_i and s_i are made by node i as a linear combination of its own variable and of the companion variables of its neighbors. However, differently from Bénézit et al. (2010)), where the communications are assumed reliable, in the case analyzed here the packets exchanged between two nodes can be lost. In this case, it is necessary to ensure that all the information sent by node i to its neighbor j is received by j at least every

once in a while.

The remarkable idea that allows to meet this requirement is that of introducing the use of counters: in particular node i has a counter $\sigma_{i,q}(k)$ ($\sigma_{i,s}(k)$ respectively) to keep track of the total q-mass (total s-mass) [3] sent by itself to its neighbors from time 0 to time k, while node j has a counter $\rho^{(i)}_{j,q}(k)$ ($\rho^{(i)}_{j,s}(k)$ resp.) to take into account the total q-mass (total s-mass) received from its neighbor i from time 0 to time k (one such variable for all $i \in \mathcal{N}_j^{\text{in}}$).

Using these mass counters, if at time k node j receives information from node i, the information coming from node i used in the update of the variable $q_j(k)$ ($s_j(k)$ resp.) will be $\sigma_{i,q}(k) - \rho^{(i)}_{j,q}(k)$ ($\sigma_{i,s}(k) - \rho^{(i)}_{j,s}(k)$ resp.); in this way the information sent by an agent but not received due to packet losses is only delayed and not lost.

The idea of using counters is inherited from the algorithm in Vaidya et al. (2011). The ra-AC algorithm, taking inspiration from the latter ideas, carries out a ratio consensus according to an asynchronous communication protocol, and the generic k-th iteration is described in Algorithm 5.1. To be implemented, each node $i \in \{1, \dots, N\}$ in the network has to keep in memory the following scalar quantities:

$$q_i(k), \quad s_i(k), \quad \sigma_{i,q}(k), \quad \sigma_{i,s}(k) \quad \rho^{(i)}_{j,q}(k), \quad \rho^{(i)}_{j,s}(k), \quad \forall (i,j) \in \mathcal{E},$$

while the quantity of interest, $x_i(k)$, is evaluated as $q_i(k)/s_i(k)$. Variables $q_i(k)$ and $s_i(k)$ are collected resp. in the N-dimensional vectors $\boldsymbol{q}(k)$ and $\boldsymbol{s}(k)$.

Suppose that at a given iteration node i wakes up. Then, the main steps of ra-AC are the following: first node i updates its variables q_i and s_i dividing their previous value by the cardinality of its out-neighbors set augmented by 1 (steps 2-3). Note that this operation leaves in fact unchanged the value of variable x_i. Then it updates the counters $\sigma_{i,q}$ and $\sigma_{i,s}$ (steps 5-6) and sends these updated values to its out-neighbors. Now, if node $j \in \mathcal{N}_i^{\text{out}}$ receives the packet from node i, it updates the variables q_j and s_j as described in steps 10-11, then it adjourns x_j and it finally stores in memory the new values for $\rho^{(i)}_{j,q}$ and $\rho^{(i)}_{j,s}$ (steps 11-12-13).

Remark 5.9.2. In case the quantity v_i in (5.25) are matrices, $v_i \in \mathbb{R}^{n \times m}$, the Algorithm 5.1 remains the same. In particular, the quantities s_i are still scalars and the ratios in Lines 5,7 and 13 are (standard) ratios between a matrix and a scalar quantity

In order to prove convergence it is necessary to introduce some assumptions on the communication. Again, the assumptions 2.3.1 and 2.3.2 introduced in Chapter 2 are used.

[3] As in Vaidya et al. (2011), the words mass and information are used interchangeably, since the physical idea of the transferring of mass quantities can be helpful in understanding how the algorithm works.

Algorithm 5.1 ra-AC Algorithm (node i)

Initialization

$q_i(0) = v_i$;
$s_i(0) = 1$;
$\sigma_{i,q}(0) = \sigma_{i,s}(0) = 0$;
$\rho_{i,q}^{(j)}(0) = \rho_{i,s}^{(j)}(0) = 0, \quad j \in \mathcal{N}_i^{\text{in}}$;

For each time k when node i wakes up

$q_i(k+1) = \frac{q_i(k)}{|\mathcal{N}_i^{\text{out}}|+1}$;
$s_i(k+1) = \frac{s_i(k)}{|\mathcal{N}_i^{\text{out}}|+1}$;
$x_i(k+1) = \frac{q_i(k+1)}{s_i(k+1)}$;
$\sigma_{i,q}(k+1) = \sigma_{i,q}(k) + q_i(k+1)$;
$\sigma_{i,s}(k+1) = \sigma_{i,s}(k) + s_i(k+1)$;

Node i broadcasts variable $\sigma_{i,q}(k+1)$ and $\sigma_{i,s}(k+1)$ to all $j \in \mathcal{N}_i^{\text{out}}$

if node j receives $\sigma_{i,q}(k+1)$ and $\sigma_{i,s}(k+1)$ **then**
    $q_j(k+1) = q_j(k) + \sigma_{i,q}(k+1) - \rho_{j,q}^{(i)}(k)$;
    $s_j(k+1) = s_j(k) + \sigma_{i,s}(k+1) - \rho_{j,s}^{(i)}(k)$;
    $x_j(k+1) = \frac{q_j(k+1)}{s_j(k+1)}$;
    $\rho_{j,q}^{(i)}(k+1) = \sigma_{i,q}(k+1)$;
    $\rho_{j,s}^{(i)}(k+1) = \sigma_{i,s}(k+1)$;
end if

The variables of the other nodes are not changed

According to these assumptions, each agent $i \in \mathcal{V}$ receives information coming from each agent $j \in \mathcal{N}_i^{\text{in}}$ at least once within any window of $h\tau$ iterations of the algorithm.

Remark 5.9.3. The assumption that two agents which are supposed to communicate have in fact to communicate within a finite time window, is standard in the context of consensus-based algorithms with directed graphs. It is necessary to guarantee deterministic exponential convergence as shown in Moreau (2005), in the sense that if this do not hold, it is possible to construct a sequence of message exchange (among all agents) that do not guarantee exponential convergence.

The following Theorem shows that, with a proper initialization of the variables, the ra-AC algorithm works as an average consensus algorithm, that is, the variables x_i, $i \in \{1, \dots, N\}$, which are updated distributively and iteratively, converge to the average of the N components of the vector $\boldsymbol{v}$.

Theorem 5.9.4. *Under Assumptions 2.3.2 and 2.3.1 and under the following initializa-*

tion for the variables

$$\begin{aligned}
&\boldsymbol{q}(0) = \boldsymbol{v}, \qquad \boldsymbol{s}(0) = \mathbf{1}_N, \\
&\sigma_{i,q}(0) = \sigma_{i,s}(0) = 0, \ \forall i \{1, \dots, N\}, \\
&\rho^{(i)}_{j,q}(0) = \rho^{(i)}_{j,s}(0) = 0, \ \forall (i,j) \in \mathcal{E},
\end{aligned}$$

the evolution, obtained using ra-AC algorithm, of the variable $\boldsymbol{x}(k)$ *exponentially converges to* $\bar{v}\mathbf{1}_N$, $\bar{v} = \boldsymbol{v}^\top \mathbf{1}_N/N$, *that is, there exist suitable constants* $C > 0$, $0 < d < 1$ *such that*

$$\|\boldsymbol{x}(k) - \bar{v}\mathbf{1}_N\|^2 \leq C \left(d^{\frac{1}{\tau_{\max}}}\right)^k \|x(0) - \bar{v}\mathbf{1}_N\|^2, \tag{5.26}$$

where $\tau_{\max} = Nh\tau$.

The complete proof can be found in Appendix D.4. However, to give an idea of the instruments used in the proof, and to introduce some quantities that will be useful in Chapter 6, the main steps of the proof are also reported in Section 5.10. A reader interested in all the passages of the proof can look at the demonstration in Appendix D.4 and then move directly to Section 5.11 (skipping Section 5.10).

Before describing the ideas behind the proof, some discussions are in order. The bound in (5.26) depends on the communication scenario through $\tau_{\max}$. For a fixed number of nodes N, $\tau_{\max}$ might increase, either because each node wakes up less often, or because each communication link may fail for a longer period of time (or both), which implies that the dissemination of information may become more difficult. In Equation (5.26), if $\tau_{\max}$ increases the upper bound becomes larger and larger, which is coherent with the fact that the information is spread through the network in a slower way.

5.10 Proof of convergence

The proof of Theorem 5.9.4 is based on the theory of ergodic coefficients for positive matrices Seneta (2006), applied to the particular case of stochastic matrices. The proof first follows what is done in Vaidya et al. (2011). However, the Assumptions 2.3.2 and 2.3.1 allow to state the results in Vaidya et al. (2011) without resorting to probability theory. In particular, ergodicity theory is exploited in such a way that the exponential convergence of the algorithm can be proven.

To proceed with the proof, first a matrix form for the algorithm is introduced, then the properties of the matrices involved are studied and finally ergodicity theory is used to prove the convergence of the algorithm.

Matrix form and properties

The matrix form for the algorithm is obtained using an **augmented state**. Namely, the evolution of variables q_i and s_i is described with the help of some additional variable, which too evolve in time.

In particular, it is necessary to introduce, for all $(i,j) \in \mathcal{E}$, the following variables

$$\nu_{j,q}^{(i)}(k) = \sigma_{i,q}(k) - \rho_{j,q}^{(i)}(k),$$
$$\nu_{j,s}^{(i)}(k) = \sigma_{i,s}(k) - \rho_{j,s}^{(i)}(k),$$

which can be collected in the column vectors $\boldsymbol{\nu}_q(k) = [\nu_{j,q}^{(i)}(k)] \in \mathbb{R}^E$, $\boldsymbol{\nu}_s(k) = [\nu_{j,s}^{(i)}(k)] \in \mathbb{R}^E$ respectively. Note that each $\nu_{j,q}^{(i)}(k)$ contains the remaining information at time k sent by node i which has not yet reached node j (due to packet losses).

Defining the row vectors $\boldsymbol{q}_a(k) = [\boldsymbol{q}(k)^\top \ \boldsymbol{\nu}_q(k)^\top]$ and $\boldsymbol{s}_a(k) = [\boldsymbol{s}(k)^\top \ \boldsymbol{\nu}_s(k)^\top] \in \mathbb{R}^{N+E}$, it is possible to show that there exists a sequence of matrices $M(k) \in \mathbb{R}^{(N+E)\times(N+E)}$ according to which it holds

$$\begin{cases} \boldsymbol{q}_a(k+1) = \boldsymbol{q}_a(k)M(k) \\ \boldsymbol{s}_a(k+1) = \boldsymbol{s}_a(k)M(k) \end{cases}. \tag{5.27}$$

Vectors $\boldsymbol{q}_a(k)$ and $\boldsymbol{s}_a(k)$ represent the augmented states. Each matrix $M(k)$ depends on the node that wakes up at time k and on which transmissions are successful at the same time step. In any case, all matrices $M(k)$, which are gathered together in the matrix set $\mathcal{M}$, satisfy the following lemma.

Lemma 5.10.1. *The set of matrices $\mathcal{M}$ satisfies*

1. *$\mathcal{M}$ is a finite set;*
2. *each $M \in \mathcal{M}$ is a row-stochastic matrix;*
3. *each positive element in any matrix $M \in \mathcal{M}$ is lower bounded by a positive constant c;*
4. *given $\tau_{\max} = Nh\tau$, $\forall k \geq 0$, the stochastic matrix*

$$V^{(\tau_{\max})}(k) = M(k)M(k+1)\cdots M(k+\tau_{\max}-1), \ M(t) \in \mathcal{M},$$

is such that its first N columns have all the elements which are strictly positive.

Remark 5.10.2. The constant $\tau_{\max}$ has been evaluated in the worst possible scenario, in particular assuming that in graph $\mathcal{G}$ there are at least two nodes that communicate with

each other in no less than $N-1$ steps. It was also assumed that the communication along one link fails $L-1$ times consecutively. In a random network $\mathcal{G}$, where the diameter of the graph is usually much smaller than the number of nodes, the actual constant $\tau_{\max}$, according to which the first N columns of $V^{(\tau_{\max})}(k)$ are strictly positive, will be, in general, much smaller.

Ergodicity theory and convergence of ra-AC

First some useful concepts of ergodicity theory will be briefly recalled. An exhaustive explanation for ergodicity theory can be found in Seneta (2006).
Given a stochastic matrix $P \in \mathbb{R}^{N\times N}$, a coefficient of ergodicity for P quantifies how much its rows are different from each other. Two well-known coefficients of ergodicity for a stochastic matrix P are

$$\delta(P) := \max_j \max_{i_1,i_2} |[P]_{i_1 j} - [P]_{i_2 j}|\,,$$
$$\lambda(P) := 1 - \min_{i_1,i_2} \sum_j \min\{[P]_{i_1 j}, [P]_{i_2 j}\}\,.$$

As all the coefficients of ergodicity, it holds that $0 \le \delta(P) \le 1$ and $0 \le \lambda(P) \le 1$.
Consider now a stochastic matrix P such that $\delta(P) < \psi$. Selecting two elements in any column of P, the difference between these two elements is necessarily smaller than ψ.
Consider now a vector $\boldsymbol{q} \in \mathbb{R}^N$ which sums to 0, that is $\mathbf{1}_N^\top \boldsymbol{q} = 0$. Define the related quantities

$$q_{\text{pos}} = \sum_{i|y_i>0} q_i \ge 1, \quad q_{\text{neg}} = \sum_{i|y_i<0} q_i \le 0, \quad q_{\text{pos}} + q_{\text{neg}} = 0,$$

and suppose that[4] $q_{\text{pos}} > 0$. It is possible to show that

$$\left|[\boldsymbol{q}^\top P]_j\right| \le \psi \sum_{i=1}^N |q_i| \tag{5.28}$$

An important property that holds for $\delta(\cdot)$ and $\lambda(\cdot)$ is the following: given r stochastic matrices $P_1, \dots, P_r$, then

$$\delta(P_1 P_2 \cdots P_r) \le \prod_{i=1}^r \lambda(P_i). \tag{5.29}$$

A stochastic matrix P such that $\lambda(P) < 1$ is called scrambling, and a sufficient condition for P to be scrambling is that at least one column is strictly positive,as can be verified by the definition of $\lambda(\cdot)$.

[4]The bound in Equation 5.28, is still verified if $q_i = 0\ \forall i$.

It is possible to apply this theory to the forward product of matrices that define the evolution of the algorithm as seen in (5.27), that is to matrix $T(k) = M(0)M(1)\cdots M(k)$. According to this matrix, it holds that $\boldsymbol{q}_a(k+1) = \boldsymbol{q}_a(0)T(k)$. First define

$$W(r) = \prod_{k=(r-1)\tau_{\max}}^{r\tau_{\max}-1} M(k),\ h \geq 1,\ M(k) \in \mathcal{M}$$

which, by Lemma 5.10.1, has strictly positive columns. As a consequence, $\lambda(W(r)) < 1$ for all $r \geq 1$. The number of different $W(r)$ is finite and, collecting all $W(r)$ in set $\mathcal{W}$, it is possible to define value $d = \max_{W\in\mathcal{W}} \lambda(W)$, $d < 1$. Due to Formula 5.29, for big enough k, $\delta(T(k)) < 1$ and in particular the following lemma holds

Lemma 5.10.3. *The constant $\beta = d^{1/(2\tau_{\max})}$, $0 < \beta < 1$, is such that $\delta(T(k)) \leq \beta^k$ for $k \geq \tau_{\max}$.*

Lemma 5.10.3 implies that the coefficient of ergodicity for $T(k)$ converges to 0 as k goes to infinity.
It is now possible to finally prove convergence in case vector $\boldsymbol{v}$ is zero mean, $\bar{v} = 0$. For $k \geq \tau_{\max}$, by Lemma 5.10.3 it holds $\delta(T(k)) \leq \beta^k$. Starting from Formula (5.28), and remembering that the first N elements of $\boldsymbol{q}_a(0)$ are $\boldsymbol{q}(0)$ and the other elements are 0, some algebraic manipulation leads to [5]

$$|x_i(k+1)| = \left|\frac{q_i(k+1)}{s_i(k+1)}\right| \leq \frac{1}{s_i(k)}\beta^k \sum_i |q_i| \leq \frac{1}{\mu}\beta^k \sum_i |q_i|$$

and then to

$$\|\boldsymbol{x}(k+1)\|^2 \leq \frac{N}{\mu^2\beta^2}(\beta^2)^{k+1}\left(\sum_i |q_i|\right)^2 \leq \frac{N^2}{\mu^2\beta^2}(\beta^2)^{k+1}\|\boldsymbol{x}(0)\|^2.$$

It is then possible to prove that, using constant $C = N^2/(\mu^2 d)$, the inequality holds for all k, that is

$$\|\boldsymbol{x}(k)\|^2 \leq C(d^{1/\tau_{\max}})^k \|\boldsymbol{x}(0)\|^2,\ k \geq 0. \tag{5.30}$$

The exponential convergence of the algorithm when vector $\boldsymbol{v}$ is 0-mean has been proven, since vector $\boldsymbol{x}(k)$ is converging to $0\mathbf{1}_N$.
This convergence result can be generalized to the case in which $\boldsymbol{v}$ is such that $\bar{v} \neq 0$, obtaining

$$\|\boldsymbol{x}_{\bar{v}}(k) - \bar{v}\mathbf{1}_N\|^2 \leq C(d^{1/\tau_{\max}})^k \|\boldsymbol{x}_{\bar{v}}(0) - \bar{v}\mathbf{1}_N\|,\ k \geq 0.$$

[5] It is possible to show that $s_i(k) > c^{\tau_{\max}}$ for all i and k.

Remark 5.10.4. If no packet losses occur, the variables $\sigma_{i,q}(k)$, $\sigma_{i,s}(k)$, $\rho_{i,q}^{(j)}(k)$ and $\rho_{i,s}^{(j)}(k)$ can be discarded (and variables q_i and s_i of the nodes that receive the information are updated directly using the packets they receive). The algorithm obtained in this case is subsumed by those presented in (Bénézit et al., 2010). The idea of using a consensus algorithm with an augmented state in order to prove the convergence of this particular ratio consensus is taken from Vaidya et al. (2011). However, in the latter the communication is synchronous, that is at each iteration all the nodes perform some updates, and moreover the results concerning the convergence are given in probability. In the set-up considered here, the algorithm is asynchronous and the convergence result is stated considering a worst-case scenario. This is a consequence of the two assumptions done on the communication, which, remarkably, also allow to prove that the convergence is exponential.

5.11 Simulations for the ra-AC algorithm

This section shows the results of some simulations done for ra-AC, and also some comparison with PDMM, which is the only other algorithm (to the best of the author's knowledge) provably convergent in case of lossy communication Sherson et al. (2017). The first simulations concern only the ra-AC. The set-up of these simulations is the following: the number of agents considered is $N = 50$, the underlying communication graph is random geometric with distance threshold equal to $r > 0$. In addition, in order to work on directed graphs, some of the links have been forced to be unidirectional. The value of τ and h for Assumptions 2.3.2 and 2.3.1 are respectively 75 and 10. In particular, different probability p of losing a given packet are considered, but if the link that is selected has failed to transmit for $h-1$ previous consecutive times, then the link is forced to be reliable without considering the packet loss probability. In Table 5.1 the averaged root mean squared error (ARMSE) of the results are given. In particular, for each value of r and p selected, $M = 500$ Monte Carlo runs (MCR) for different graph realizations are carried out. Denoting with $\boldsymbol{x}_{\{i\}}(k)$ the value $\boldsymbol{x}(k)$ obtained in the $i-$th MCR, then

$$\text{ARMSE}(k) = \frac{1}{M}\sum_{i=1}^{M}\left[\frac{1}{\sqrt{N}}\|\boldsymbol{x}_{\{i\}}(k) - \bar{v}\mathbf{1}_N\|_2\right]$$

The results of the simulations show that the more connected the graph is, the faster the convergence is. On the other hand, the packet loss probability, as expected, makes the convergence slower. Note that for $r = 0.25$, even at iteration 2000 the convergence is still not good. However, even in the best case, at iteration 2000 all the nodes have woken

ARMSE(2000)	$p = 20\%$	$p = 50\%$	$p = 80\%$
$r = 0.25$	0.395	0.635	1.22
$r = 0.33$	0.033	0.131	0.45
$r = 0.5$	$1.62\,10^{-5}$	$9.17\,10^{-4}$	0.0319

Table 5.1: Values of $ARMSE$ for the ra-AC algorithm at time $k = 2000$, computed over $M = 500$ Monte Carlo runs for different values of r and p.

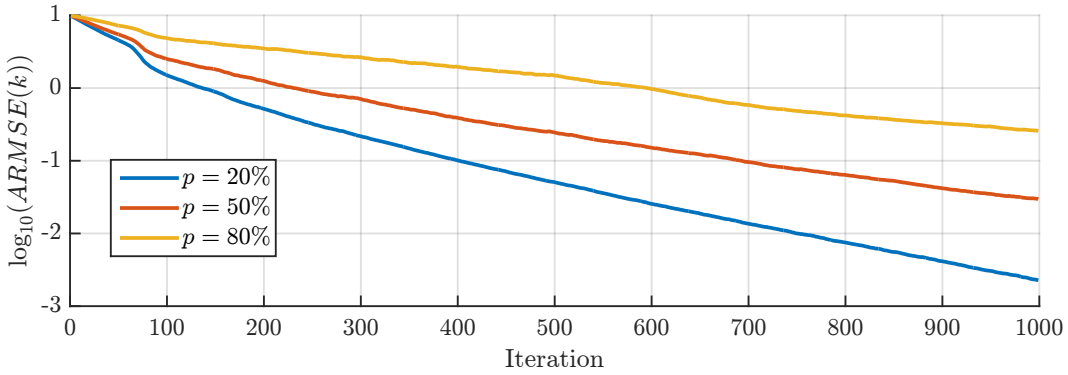


Figure 5.5: RMSE as a function of time for 3 different values for the packet loss probability, evaluated over $M = 500$ MCR. The value for the maximum distance r between nodes is 1/3.

up at most 40 times, and so, due to the presence of packet losses and the fact that each node have only few neighbors, this is not surprising.
Figure 5.5 shows the time evolution for the ARMSE(k), in case $r = 1/3$. For all the different values of the packet loss probability it is possible to appreciate the exponential convergence of the algorithm.

Finally, Figure 5.6 shows the time evolution of the variables of a single node in the network. It is interesting to see that, while the ratio between $q_{14}(k)$ and $s_{14}(k)$ exponentially converges to the mean $\bar{v}$, the single variables do not converge but keep oscillating. This behavior is typical in the ratio consensus algorithm in presence of unidirectional links and packet losses.

The second part of the simulations are done for comparison. The algorithms used are ra-AC and PDMM Sherson et al. (2017). The communication graph is again random geometric with $N = 50$ nodes, and it is left undirected. The distance threshold r is 0.3 for Figure 5.7, and 0.9 for Figure 5.8. In this case, the figures represent a single simulation (there is no average on different MCR). The parameter τ concerning a minimum frequency of activation for each node is set to 65 in these simulations, while no value for h is fixed. Concerning the unreliable communication, in fact, each exchange is unsuccessful with probability p, and successful with probability $1 - p$, but there is no guarantee that the

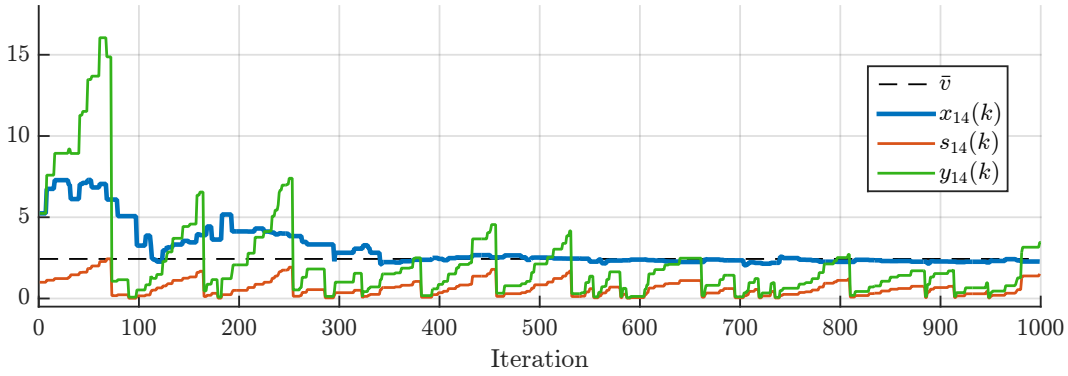


Figure 5.6: Time evolution for the scalar variables $x_{14}(k)$, $q_{14}(k)$ and $s_{14}(k)$ for one run of the algorithm. In this simulation $r = 1/3$ and $p = 20\%$.

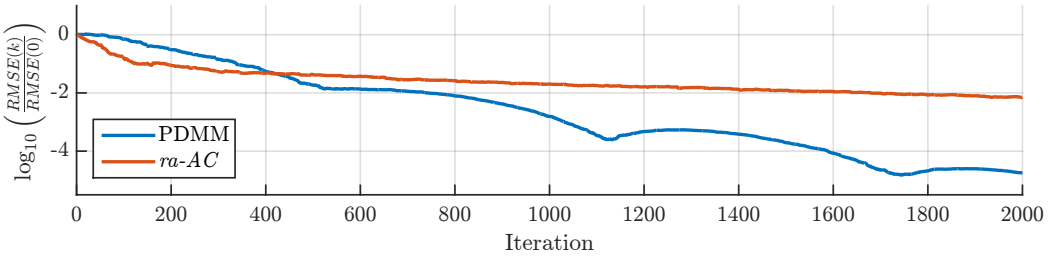


Figure 5.7: Comparison among ra-AC and PDMM. The communication graph is random geometric, with a distance threshold $r = 0.3$. The packet loss probability is 20%.

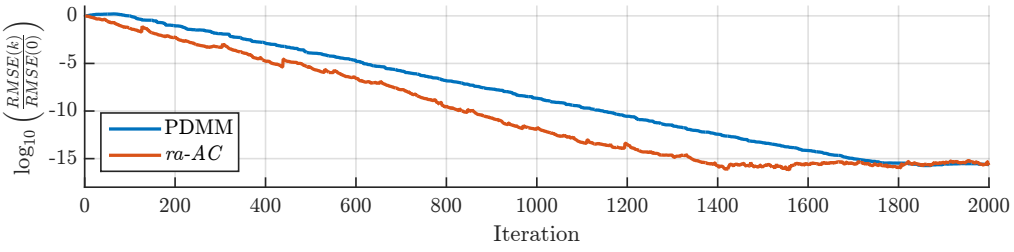


Figure 5.8: Comparison among ra-AC and PDMM. The communication graph is random geometric, with a distance threshold $r = 0.9$. The packet loss probability is 20%.

communication between agent i and j (such that $(i, j) \in \mathcal{E}$) cannot be unsuccessful for more than a finite number of times. Simulations shows that, for $p = 20\%$, it is (generally) not necessary to enforce this condition.

The figures shows that PDMM can be very effective. This algorithm is similar to the ADMM, and it is interesting to see that the convergence rate of PDMM and ra-AC behaves similarly to the comparison between the average consensus algorithm and ADMM.

As a matter of fact, when the graph is sparse PDMM is faster, but when the graph is dense, ra-AC has a better performance. Again, ra-AC has no parameter to set, while PDMM has two parameters to be set. Its convergence rate really depends on the choice of one of these parameters, $\rho > 0$ (the other, α in case of quadratic function can be set to 1), as can be inferred from Table 5.2. The table shows that the communication graph

$\log_{10}\left(\frac{RMSE(2000)}{RMSE(0)}\right)$	$\rho = 0.9$	$\rho = 0.45$	$\rho = 0.1$
$r = 0.3$	-4.4718	-1.7414	-0.5922
$r = 0.9$	-2.9170	-4.9982	-14.8301

Table 5.2: Values of $\log_{10}\left(\frac{RMSE(2000)}{RMSE(0)}\right)$ for the PDMM algorithm at time $k = 2000$, for different values of the distance threshold r and PDMM parameter ρ. The packet loss probability n the simulations is 20%.

really influences the choice of the parameter. Since in a multi-agent system the global communication network might be unknown, this strong dependence may create some problem in the setting of parameter ρ. On the other hand, using ra-AC, each agent only needs to know its neighbors to run the algorithm, and no global knowledge is required. From a practical point of view, this consideration on the parameters' tuning is quite important.

5.12 Final considerations

The first part of the chapter, devoted to the comparison between consensus based and Lagrangian based algorithm, has shown the importance of the communication graph in determining the convergence rate of distributed algorithms. The analysis clearly show that one should choose the algorithm depending on the underlying communication graph. Also, among all the analyzed algorithms, the accelerated consensus seems a good choice in almost all scenarios, especially because once matrix P is set, the convergence rate of this algorithm does not depend on the curvatures of the cost functions.

The second part of the chapter focused on the development of a robust and asynchronous algorithm to solve the consensus problem. The developed approach, ra-AC, is shown to be exponentially convergent under some deterministic assumption on the communication. The algorithm is also compared to PDMM Sherson et al. (2017), showing advantages and disadvantages of both algorithms.

6

Minimization of additively separable cost functions

This chapter partly extends the following paper (submitted to the IEEE Transactions on Automatic Control)

Bof N., Carli R., Notarstefano G., Schenato L., and Varagnolo D. Newton-Raphson Consensus under lossy communication for peer-to-peer optimization. *arXiv preprint arXiv:1707.09178*, 2017c

In this chapter, the problem analyzed is the unconstrained minimization of the sum of convex functions in a fully distributed multi-agent setting subject to a real-world communication scenario. Even though the distributed optimization of convex functions is a well studied problem, the focus of this chapter is in considering **asynchronous computation** and **lossy communication**, with the latter being a real novelty. In particular, this chapter robustify a recently proposed algorithm named Newton-Raphson Consensus by integrating it with the ra-AC algorithm presented in Section 5.9. Separation of time scales principle allows to show that under mild conditions (i.e., persistency of the agents activation and bounded consecutive communication failures) the proposed algorithm is proved to be locally exponentially stable with respect to the optimal global solution.

6.1 Introduction and state of the art

This chapter is devoted to the last optimization problem analyzed in the thesis. The tackled problem is the minimization of the sum of cost functions. Differently from the scenario presented in Chapter 4, here the private cost functions of the agents depend all an the same optimization variable. An example of such a scenario is the quadratic problem analyzed in the previous chapter. The aim of this chapter is to find an algorithm to solve more general problems than the quadratic one. Again, the main feature of the developed algorithm concerns its ability to **deal with a real communication scenario**. As in all the previous case, the algorithm has to be able to cope with unreliable communications, using a communication protocol which does not implement an acknowledge system and is also asynchronous.

The following analysis of the literature involve also some works that have been already recalled in the thesis. The analysis in Chapter 4 focused on showing how the previous works did not simultaneously met the requirement of local information exchange, asynchronous update and robustness to packet loss. In Chapter 5 the focus was on works mainly concerning the quadratic case since this was the restriction explored. In this chapter, the focus is on algorithms for global estimation problems and on the type of communication scenarios according to which their convergence can be shown.

As already pointed out in Section 4.1, there are mainly three types of algorithms to deal with the distributed optimization of sums of cost functions.

One of these types are *distributed subgradient methods*. These are a popular class of algorithms that are able to cope with asynchronous updates and time-varying graphs. They are simple to implement, can deal with non-differentiable convex cost functions, and require only the computation of local (sub)-gradients. However, these algorithms exhibit sub-linear converge rates even if the cost functions are smooth Nedić and Ozdaglar (2010); Nedić et al. (2010). Recent works based on this approach have extended these results to directed and possibly time-varying communication in both discrete-time Lin, Ren, and Song (2016); Nedić and Olshevsky (2015) and continuous-time settings Gharesifard and Cortes (2014); Kia, Cortés, and Martínez (2015). However, the use of a diminishing step-size tacitly implies that the communication is synchronous (since the step-size is designed as a function of the global time that triggers the algorithm). Moreover, the underlying assumption for guaranteeing convergence is that the transmitter nodes should know which packets are successfully transmitted. This assumption corresponds to employing communication protocols with reliable packet transmission or acknowledge mechanisms. As already pointed out, such solutions might be difficult or expensive to implement over wireless media. The recent work Lee and Nedić (2016) proposes an asynchronous

algorithm, based on random projections, in which the step-size (both diminishing and constant) is uncoordinated among agents.

The second popular class of distributed optimization algorithms relies on *dual decomposition schemes*. In this case the related literature is very large and the reader is referred to Yang and Johansson (2011) for a comprehensive tutorial. Among these algorithms, the Alternating Direction Method of Multipliers (ADMM) is one of the most employed, due to its simple distributed implementation and good convergence speed (as shown in Sections 5.6 and 5.7 in the particular case of quadratic cost functions). Substantial research has been dedicated in optimizing the free parameters of ADMM in order to obtain faster convergence rates. However, these are mainly restricted to synchronous implementations over undirected communication graphs Ghadimi, Teixeira, Shames, and Johansson (2015); Teixeira et al. (2013); Nishihara, Lessart, Recht, Packard, and Jordan (2015); Iutzeler et al. (2016). Some recent exceptions extend dual decomposition, Notarnicola and Notarstefano (2016), and ADMM, Wei and Ozdaglar (2013); Bianchi, Hachem, and Iutzeler (2016); Chang, Hong, Liao, and Wang (2016), to asynchronous scenarios with edge-based or node-based activation schemes. Some recent works have addressed the problem of random delay in the communication/updates rounds in ADMM schemes Zhang and Kwok (2014); Peng, Xu, Yan, and Yin (2016); Chang et al. (2016); however these strategies are restricted to networks with master-slave communication topologies (see Figure 2.3) and do not explicitly address packet losses.

The PDMM algorithm Sherson et al. (2017) (already used in Section 5.11), belongs to the latter class of algorithms, and, to the best of the author's knowledge, is the only distributed algorithm able to cope with an unreliable communication network without employing an acknowledge protocol. The version containing the formal proof of convergence in presence of packet losses and asynchronous updates was very recently published.

The third class of optimization algorithms, usually referred to as *Newton-based methods*, consists of strategies that exploit second-order derivatives, i.e., the Hessians of the cost functions for computing descent directions. For example in Wei, Ozdaglar, and Jadbabaie (2013a,b) the authors apply quasi-Newton distributed descent schemes to general time-varying directed graphs. Also Eisen, Mokhtari, and Ribeiro (2016) propose decentralized quasi-Newton methods which are provably convergent in an asynchronous set-up. Another approach, based on computing Newton-Raphson directions through average consensus algorithms, has been proposed in Varagnolo, Zanella, Cenedese, Gianluigi, and Schenato (2016). Even if initially proposed for synchronous implementations, this scheme has been later extended to cope with asynchronous symmetric gossip communi-

cation schemes Zanella, Varagnolo, Cenedese, Pillonetto, and Schenato (2012). Works Zanella et al. (2012) and Varagnolo et al. (2016) have introduced the idea of tracking the gradient of the whole cost function as a mean to replace the diminishing stepsize with a constant one. This idea has been reconsidered with different averaging schemes and formalized in Di Lorenzo and Scutari (2016) to handle nonconvex optimization (combined with a successive convex approximation approach) and in Nedić, Olshevsky, and Shi (2016) and Qu and Li (2017) to show linear convergence with constant stepsize. Recently, in Mansoori and Wei (2017) a Newton scheme with almost sure global-linear and local-superlinear convergence has been proposed for a different problem set-up in which consistency of the local variables is only penalized but not guaranteed.

Although there exists a large body of literature on distributed convex optimization schemes employing synchronous and asynchronous communications, only PDMM directly addressed situations where the communications are unreliable and lossy. In such a communication scenario, in fact, trying to make the other aforementioned algorithms cope with packet losses using naïve modifications (e.g., using the most recently received message from the neighboring nodes, interpretable as using delayed information in the algorithms) may destroy some of the hypotheses that guarantee the convergence of the original algorithms (e.g., the doubly stochasticity or the invariance of some quantities such as the global averages). Distributed convex optimization in the presence of lossy communications is thus a non-trivial task.

The main contribution of this chapter is to propose a set of distributed optimization algorithms which are robust to packet losses for general asynchronous peer-to-peer networks and are guaranteed to have (local) exponential convergence.

More specifically, the starting algorithm is the Newton-Raphson Consensus initially proposed in Varagnolo et al. (2016). In this distributed algorithm, the only step which requires exchange of messages is an average consensus step. As a consequence, if the aim is to robustify the algorithm to make it cope with unreliable communication, this is the step in which one has to intervene. The idea is then to employ the ra-AC algorithm developed in Section 5.9, since this algorithm is robust and moreover asynchronous.

This new scheme, together with PDMM, are (to the best of the author's knowledge) the first distributed optimization algorithms able to deal with asynchronous and lossy communication protocols.

Despite the simple intuition of combining a new consensus algorithm with the Newton-Raphson Consensus approach, the algorithm analysis required the development and application of non-standard non-linear control methods, namely ad-hoc results on exponential stability of non-linear, time-varying discrete-time systems with multiple interconnections

by means of time-scale separation. One additional element of complexity arises from the fact that some of the variables involved in the algorithm do not converge to a steady-state value but oscillate, therefore standard exponential stability proofs cannot be exploited.

Apart from the standard Newton-Raphson consensus, which requires the computation of the inverse of the Hessian of the functions, also two other computationally lighter versions are provided. They are referred to as Jacobi Consensus and Gradient Consensus. The former requires the evaluation of only a part of the Hessian and the inversion of a diagonal matrix, while the latter requires only first order information (that is only the gradient) and no matrix inversion. These versions are computationally more efficient at the price of slower convergence rate, while still guaranteeing linear convergence.

Under mild conditions, i.e., persistency of (asynchronous) node updates, uniformly bounded consecutive communication link failures, and connectivity of the communication graph, it is possible to prove the convergence of the algorithm. In particular one can show that the optimization algorithm is locally exponentially stable with respect to the global solution as long as the step-size of the updates is smaller than a certain critical value and the cost functions are sufficiently smooth. The proof is based on time-scale separation and Lyapunov theory, and extends the results in Carli, Notarstefano, Schenato, and Varagnolo (2015), where the convergence was proved only for quadratic cost functions. The theoretical results are complemented with numerical simulations based on real datasets under lossy, broadcast communication. The robust and asynchronous Newton-Raphson consensus presented in the chapter is numerically compared against two recently proposed algorithms Tsianos, Lawlor, and Rabbat (2012); Nedić et al. (2016) under the special case of asynchronous lossless communication, showing the better performance of the proposed Newton-Raphson approach. Finally, it is also compared with PDMM Sherson et al. (2017), and this (initial) comparison shows that the algorithm proposed in this chapter can be competitive with PDMM.

6.2 Problem formulation

The problem considered in this chapter is the following separable optimization one

$$\boldsymbol{x}^* := \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \sum_{i=1}^{N} f_i(\boldsymbol{x}) \tag{6.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and where the local costs $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ satisfy:

Assumption 6.2.1 (Cost smoothness)**.** Each f_i is known only to node i and is $\mathcal{C}^3$ and strongly convex, i.e., its Hessian is bounded from below, $\nabla^2 f_i(\boldsymbol{x}) \succ cI_n$ for all $\boldsymbol{x}$, with

$c > 0$ some positive scalar.

The communication among nodes is modeled via a communication graph $\mathcal{G}$ that meets the following assumption

Assumption 6.2.2. The communication graph $\mathcal{G}$ is time-invariant, **directed** and satisfy the Assumption 2.2.1 on connectivity.

The algorithm designed to solve Problem (6.1) has to have the following features:

1. *Asymptotic global estimation*: each agent's estimate of the global minimizer has to asymptotically converge to the optimal solution $\boldsymbol{x}^*$.
2. *Peer-to-peer (leaderless)*: each node's update has to consider the limited computational and memory capability available at the node itself and there is no master node among the agents. Moreover, the algorithm can only require communication between one-hop neighbors and it has to assure convergence on any communication graph $\mathcal{G}$ satisfying Assumption 6.2.2.
3. *Distributed*: the update-rule of the local variables at each node has to depend only on the variables stored by the local node and by its neighbors. No multi-hop information exchange is allowed.
4. *Asynchronous*: the algorithm has to allow the agents to perform the update step and the communication step in any moment, without any coordination among the agents.
5. *Lossy broadcast communication without ACK*: the convergence of the algorithm has to be assured even if communication is lossy and broadcast-based. No ACK mechanisms has to be employed.

To the best of authors' knowledge, only PDMM Sherson et al. (2017) can offer the same features.

6.3 Building blocks

The algorithm proposed consists of two different building blocks: i) the Newton-Raphson Consensus, proposed in Varagnolo et al. (2016) to solve problem (6.1) and ii) the robust ratio average consensus algorithm ra-AC proposed in Section 5.9.

The Newton-Raphson Consensus, proposed in Varagnolo et al. (2016) possesses the first three features mentioned above (i.e., 1, 2, and 3) but it assumes synchronous and reliable communications.

The ra-ACalgorithm on the other hand meets all the features above, but is limited in the problems it can solve (namely, just the quadratic ones).

The idea is to merge the two schemes above to design a distributed optimization algorithm that solves problem (6.1) and that exhibits all the features 1-5 above. The main challenge is showing that feature 1 holds, since the interaction between these algorithms might lead to instability unless some suitable assumptions are considered. The key mathematical machinery that will be used to this means is Lyapunov theory and separation of time-scales.

Before providing the description of the proposed algorithm, the Newton-Raphson Consensus and the related use of ra-AC are briefly described.

Newton-Raphson Consensus

The Newton-Raphson Consensus Varagnolo et al. (2016) is inspired by the Newton-Raphson's update in the standard centralized scenario[1] (again, x^+ denote $x(k+1)$)

$$\boldsymbol{x}^+ = \boldsymbol{x} - \epsilon(\nabla^2 f(\boldsymbol{x}))^{-1}\nabla f(\boldsymbol{x}).$$

This update can be rewritten as

$$\boldsymbol{x}^+ = (1-\epsilon)\boldsymbol{x} + \epsilon\Big(\nabla^2 f(\boldsymbol{x})\Big)^{-1}\Big(\nabla^2 f(\boldsymbol{x})\,\boldsymbol{x} - \nabla f(\boldsymbol{x})\Big),$$

and substituting $f(x)$ with the sum of the $f_i(x)$, the update becomes

$$\boldsymbol{x}^+ = (1-\epsilon)\boldsymbol{x} + \epsilon\underbrace{\Big(\sum_i \nabla^2 f_i(\boldsymbol{x})\Big)^{-1}}_{=:H(\boldsymbol{x})}\underbrace{\Big(\sum_i (\nabla^2 f_i(\boldsymbol{x})\,\boldsymbol{x} - \nabla f_i(\boldsymbol{x}))\Big)}_{=:\boldsymbol{g}(\boldsymbol{x})}. \tag{6.2}$$

The latter system is exponentially stable as long as the parameter $\epsilon > 0$, which is the stepsize, is chosen in a proper way. Assuming now that all agents can have a different value $\boldsymbol{x}_i$ of the estimate of $\boldsymbol{x}^*$, and mimicking the previous algorithm, the following N

[1] Note that the Newton-Raphson method coincides with the Damped Newton method described in Algorithm 2.1. To be consistent with Varagnolo et al. (2016), in this chapter it is denoted as Newton-Raphson's method

local updates are obtained:

$$\boldsymbol{x}_i^+ = (1-\epsilon)\boldsymbol{x}_i + \epsilon\Big(\underbrace{\sum_j \underbrace{\nabla^2 f_j(\boldsymbol{x}_j)}_{=:H_j(\boldsymbol{x}_j)}}_{=:\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}\Big)^{-1}\Big(\underbrace{\sum_j \underbrace{(\nabla^2 f_j(\boldsymbol{x}_j)\,\boldsymbol{x}_j - \nabla f_j(\boldsymbol{x}_j))}_{=:\boldsymbol{g}_j(\boldsymbol{x}_j)}}_{=:\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}\Big). \tag{6.3}$$

The dynamics of the N local systems is identical and exponentially stable, therefore, since they are all driven by the same forcing term[2] $\kappa(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n) = (\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N))^{-1}\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$, intuitively one expects that

$$\boldsymbol{x}_i - \boldsymbol{x}_j \to 0, \quad \forall i,j\ ,$$

which implies that all local variable will be identical. If this is the case, then the dynamics of each local system will eventually become the dynamics of a standard centralized Newton-Raphson algorithm.

This algorithm, however, requires each agent to be able to instantaneously compute the two sums $\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$, $\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$, which is obviously not possible in a distributed computation set-up. The original paper Varagnolo et al. (2016) extends the standard Newton-Raphson algorithm into a distributed scenario via the use of synchronous lossless average consensus protocols that asymptotically compute these sums, while Zanella et al. (2012) extends it to the case of asynchronous gossip-based lossless average consensus strategies.

How ra-AC is exploited

In Formula (6.3) it is possible to identify the presence of two average consensus. In particular, one can see the quantity $(\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N))^{-1}\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$ as

$$\left(\frac{\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}{N}\right)^{-1}\frac{\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}{N}.$$

The idea is to evaluate these 2 averages using two (slightly modified) ra-AC algorithms running in parallel, one to find the mean of $\boldsymbol{g}_1(\boldsymbol{x}_1),\dots,\boldsymbol{g}_N(\boldsymbol{x}_N)$ and the other to determine the mean of $H_1(\boldsymbol{x}_1),\dots,H_N(\boldsymbol{x}_N)$. Looking at Algorithm 5.1, the two ra-AC algorithms to evaluate the two means only differ in the initialization of q_i in Step 1. To distinguish between the two different runs, the variables q_i of the first ra-AC are called $\boldsymbol{y}_i$, while the variables q_i of the second ra-AC are called Z_i instead[3]. In particular, the initialization of

[2] Assumption 6.2.1 assures that $\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$ is invertible.

[3] Note that the first ra-AC algorithm performs the mean of a group of vectors, the $\boldsymbol{g}_i(\boldsymbol{x}_i)$, while the second performs the mean of matrices, the $H_i(\boldsymbol{x}_i)$.

the $\boldsymbol{y}_i$ and Z_i is as follows

$$\begin{aligned}\boldsymbol{y}_i(0) &= \boldsymbol{g}_i(\boldsymbol{x}_i), \quad i \in \mathcal{V},\\ Z_i(0) &= H_i(\boldsymbol{x}_i), \quad i \in \mathcal{V}.\end{aligned}$$

According to the ra-AC algorithm, the $i-$th agent's estimate at time k of the first mean is obtained as the ratio of $\boldsymbol{y}_i(k)$ and $s_i(k)$, and the estimate of the second as the ratio of $Z_i(k)$ and $s_i(k)$ (the $s_i(k)$ is the same for the two ra-AC, since the agents' activation sequence is the same for the two algorithms and it is assumed that the messages concerning the two different algorithms are sent together, and so are both received or both lost). If the Assumptions 2.3.1 and 2.3.2 hold, due to the convergence property of ra-AC it holds

$$\lim_{k\to\infty} \frac{y_i(k)}{s_i(k)} = \frac{\overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}{N}, \quad \lim_{k\to\infty} \frac{Z_i(k)}{s_i(k)} = \frac{\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)}{N}, \quad \forall\, i \in \mathcal{V}.$$

Due to the properties of the limits, it also holds

$$\lim_{k\to\infty} \left(\frac{Z_i(k)}{s_i(k)}\right)^{-1} \frac{y_i(k)}{s_i(k)} = \left(\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)\right)^{-1} \overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N), \quad \forall\, i \in \mathcal{V}.$$

The left-hand side of the latter equation can be rewritten simplifying $s_i(k)$, finally obtaining

$$\lim_{k\to\infty} (Z_i(k))^{-1}\, y_i(k) = \left(\overline{H}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)\right)^{-1} \overline{\boldsymbol{g}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N), \quad \forall\, i \in \mathcal{V}.$$

Remark 6.3.1. The results on the convergence of the proposed algorithm are local and for this case it is possible to guarantee that $Z_i(k) \succ cI_n$, for all times k. As a consequence it is always possible to evaluate the inverse of $Z_i(k)$.

Since the $s_i(k)$s simplify in the ratio of the two means, these variables are not used in the algorithm presented in the next section.

6.4 The robust asynchronous Newton-Raphson Consensus (ra-NRC)

This section merges the *Newton-Raphson Consensus* and the ra-AC algorithm into one algorithm, called robust asynchronous Newton-Raphson Consensus (ra-NRC). This new algorithm solves problem (6.1) and exhibits all the features listed in Section 6.2. It can be organized in a block scheme as in Figure 6.1.

In the following a "meta distributed algorithm" is proposed. It can result in different

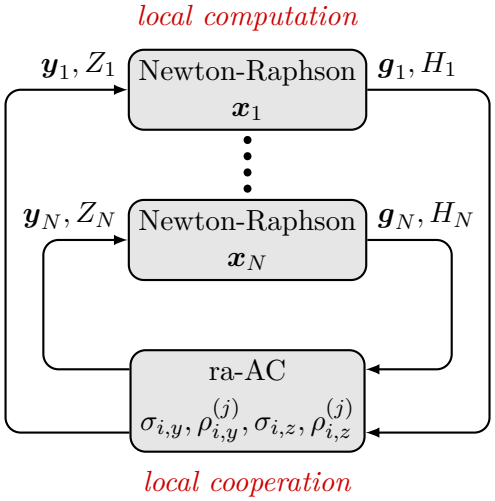


Figure 6.1: Graphical representation of the robust asynchronous Newton-Raphson Consensus (ra-NRC).

distributed algorithms depending on the (possibly asynchronous and lossy) communication protocol implemented in the network.

The meta algorithm consists of four main blocks of code implemented by each node $i \in \mathcal{V}$ in the network: *Initialization* (at startup), *Data Transmission*, *Data Reception* and *Estimate Update*.

Except for the first block, which corresponds to a one-time execution at startup, the blocks can be executed asynchronously, with possibly different execution rates. The scheduling of these three blocks, for each agent i, is determined by three binary variables $\texttt{flag}_{\texttt{transmission},i}$, $\texttt{flag}_{\texttt{reception},i}$, and $\texttt{flag}_{\texttt{update},i}$, whose evolution is determined by the communication protocol. Each code block is assumed to be executed *sequentially* and *atomically*, i.e., the local variables and flags cannot be changed by any other process. For example, if a node is executing *Estimate Update* and a new packet is incoming, this packet is either dropped or placed in a buffer till *Estimate Update* is not completed. Thus, a distributed algorithm will be simply the combination of the given meta scheme with a communication protocol defining how the flags are activated. For example, in an event-triggered communication protocol the reception of a packet may sequentially trigger (if no other block is being executed) the *Data Reception* block, which then triggers the *Estimate Update* block, and that finally triggers the *Data Transmission* block. In the following it is assumed that when an agent is idle, it is always ready to receive a new packet and when a packet is received by the i-th node then $\texttt{flag}_{\texttt{reception},i}$ is set to one.

One of the strengths of the proposed algorithm, is that its convergence is independent of the specific communication protocol as long as this protocol satisfies some mild

assumptions in terms of minimum scheduling rate of each block and maximum consecutive packet losses, as formally stated in the next section.

Algorithm 6.1 provides a pseudo-code description of ra-NRC.

The first block *Initialization* (lines 1-7) is a one-time operation preformed by each node at the beginning of the algorithm. The only free parameter to set is the initial estimate $\boldsymbol{x}^o$ for the global optimization, while all other variables depend on this choice for $\boldsymbol{x}^o$.

The blocks *Data Transmission* (lines 8-16) and *Data Reception* (lines 17-25) implement the ra-AC algorithm (see bottom block in Figure 6.1). Note that, as written in Algorithm 6.1, the ra-AC is *fully parallel*, in the sense that multiple nodes can transmit at the same time, since any potential collision will result in a packet loss already handled by the algorithm.

Specifically, when node i is performing the *Data Transmission* block, the updates of variables $\boldsymbol{y}_i, Z_i$ (line 10-11) correspond to the update of q_i in Algorithm 5.1, line 5. The update for $\sigma_{i,y}$ (line 12) corresponds to the update of $\sigma_{i,q}$ in Algorithm 5.1 (line 8)(and the same holds for the update of $\sigma_{i,z}$ (line 13)). After computing $\sigma_{i,y}$ and $\sigma_{i,z}$, the transmitting node broadcasts $\sigma_{i,y}, \sigma_{i,z}$ and its ID to its neighbors. After transmission, the node returns to an idle-mode (line 15). When node i is in the receiving mode and it receives a message (line 17), it extracts the transmitter node ID j and the corresponding variables $\sigma_{j,y}, \sigma_{i,z}$ (line 18). The variable $\boldsymbol{y}_i$ is updated like variable q_j in Algorithm 5.1 (line 11) and then the local variable $\rho_{i,y}^{(j)}$ is finally updated (line 21) similarly to the update in line 14 of Algorithm 5.1 (the same comparisons can be made for lines 20 and 22 of Algorithm 6.1 regarding variables Z_i and $\rho_{i,z}^{(j)}$).

The last block *Estimate Update* is responsible for implementing a local version of the Newton-Raphson method. The update of the local estimate $\boldsymbol{x}_i$ of the global optimizer, available at each node i, is performed via the Newton-Raphson Consensus described in the previous section. In practice, the roles of $\boldsymbol{y}_i$ and Z_i are those of (scaled) local approximations of the global functions $\overline{\boldsymbol{g}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and $\overline{H}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ defined above. As so, mimicking Eqn. (6.3), the proposed algorithm uses these variables to implement an approximated Newton-Raphson (line 27). Since the local variables $\boldsymbol{x}_i$ are continuously updated, also the global functions $\overline{\boldsymbol{g}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_i \boldsymbol{g}_i(\boldsymbol{x}_i)$ and $\overline{H}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_i H_i(\boldsymbol{x}_i)$ need to be updated accordingly. This cannot be done instantaneously due to the networked nature of the framework and has be achieved through the ra-AC (see Figure 6.1). In order to be able to track the continuously changing signals $\boldsymbol{g}_i$ and H_i, each node has to compute these signals before and after updating the $\boldsymbol{x}_i$ ($\boldsymbol{g}_i^{\text{old}}$ e H_i^{old} in lines (28-29) and $\boldsymbol{g}_i$ and H_i lines (30-31), respectively)

Algorithm 6.1 robust asynchronous Newton-Raphson Consensus (ra-NRC) for node i

Require: x^o, ϵ, c

Initialization (atomic)

$\boldsymbol{x}_i \leftarrow \boldsymbol{x}^o$

$\boldsymbol{y}_i \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)\boldsymbol{x}^o - \nabla f_i(\boldsymbol{x}^o)$, $\boldsymbol{g}_i \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)\boldsymbol{x}^o - \nabla f_i(\boldsymbol{x}^o)$, $\boldsymbol{g}_i^{old} \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)\boldsymbol{x}^o - \nabla f_i(\boldsymbol{x}^o)$

$Z_i \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)$, $H_i \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)$, $H_i^{old} \leftarrow \nabla^2 f_i(\boldsymbol{x}^o)$

$\sigma_{i,y} \leftarrow \mathbf{0}$, $\sigma_{i,z} \leftarrow \mathbf{0}$

$\rho_{i,y}^{(j)} \leftarrow \mathbf{0}$, $\rho_{i,z}^{(j)} \leftarrow \mathbf{0}$, $\forall j \in \mathcal{N}_i^{\text{in}}$

$\texttt{flag}_{\texttt{reception},i} \leftarrow 0$, $\texttt{flag}_{\texttt{update},i} \leftarrow 0$

$\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$

Data Transmission (atomic)

if $\texttt{flag}_{\texttt{transmission},i} = 1$ **then**

  $\texttt{transmitter\_node\_ID} \leftarrow i$

  $\boldsymbol{y}_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\boldsymbol{y}_i$

  $Z_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} Z_i$

  $\sigma_{i,y} \leftarrow \sigma_{i,y} + \boldsymbol{y}_i$

  $\sigma_{i,z} \leftarrow \sigma_{i,z} + Z_i$

  Broadcast: $\texttt{transmitter\_node\_ID}, \sigma_{i,y}, \sigma_{i,z}$

  $\texttt{flag}_{\texttt{transmission},i} \leftarrow 0$

end if

Data Reception (atomic)

if $\texttt{flag}_{\texttt{reception},i} = 1$ and a message is received **then**

  $j \leftarrow \texttt{transmitter\_node\_ID}$, $(j \in \mathcal{N}_i^{\text{in}})$

  $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i + \sigma_{j,y} - \rho_{i,y}^{(j)}$

  $Z_i \leftarrow Z_i + \sigma_{j,z} - \rho_{i,z}^{(j)}$

  $\rho_{i,y}^{(j)} \leftarrow \sigma_{j,y}$

  $\rho_{i,z}^{(j)} \leftarrow \sigma_{j,z}$

  $\texttt{flag}_{\texttt{reception},i} \leftarrow 0$

  $\texttt{flag}_{\texttt{update},i} \leftarrow 1$ (optional)

end if

Estimate Update (atomic)

if $\texttt{flag}_{\texttt{update},i} = 1$ **then**

  $\boldsymbol{x}_i \leftarrow (1-\epsilon)\boldsymbol{x}_i + \epsilon {Z_i}^{-1}\boldsymbol{y}_i$

  $\boldsymbol{g}_i^{\text{old}} \leftarrow \boldsymbol{g}_i$

  $H_i^{\text{old}} \leftarrow H_i$

  $H_i \leftarrow \nabla^2 f_i(\boldsymbol{x}_i)$

  $\boldsymbol{g}_i \leftarrow H_i\boldsymbol{x}_i - \nabla f_i(\boldsymbol{x}_i)$

  $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i + \boldsymbol{g}_i - \boldsymbol{g}_i^{\text{old}}$

  $Z_i \leftarrow Z_i + H_i - H_i^{\text{old}}$

  $\texttt{flag}_{\texttt{update},i} \leftarrow 0$

  $\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$ (optional)

end if

and then update the "consensus" variables $\boldsymbol{y}_i$ and Z_i in order to track the current sums $\overline{\boldsymbol{g}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$ and $\overline{H}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$ (lines 32-33). In fact, this operation guarantees that the following relationships hold preserved:

$$\sum_i (\boldsymbol{y}_i(k) + \sum_{j \in \mathcal{N}_i^{\text{in}}} (\sigma_{j,y}(k) - \rho_{i,y}^{(j)}(k))) = \sum_i \boldsymbol{g}_i(k), \tag{6.4}$$

$$\sum_i (Z_i(k) + \sum_{j \in \mathcal{N}_i^{\text{in}}} (\sigma_{j,z}(k) - \rho_{i,z}^{(j)}(k))) = \sum_i H_i(k), \tag{6.5}$$

where, with a slight abuse of notation, with $\boldsymbol{g}_i(k)$ and $H_i(k)$ we denote $\boldsymbol{g}_i(\boldsymbol{x}_i(k))$ and $H_i(\boldsymbol{x}_i(k))$ respectively (Lemma 6.5.3 shows this property in a particular set-up). The intuition behind the convergence of the algorithm is that, if the local estimates $\boldsymbol{x}_i$ change slower than the rate at which the ra-AC converges, which can be achieved by choosing a sufficiently small stepsize ϵ, then one would expect that

$$\lim_{k \to \infty} (Z_i(k))^{-1} \boldsymbol{y}_i(k) = \left(\sum_i H_i(k) \right)^{-1} \sum_i \boldsymbol{g}_i(k). \tag{6.6}$$

A formal proof of the ra-NRC algorithm and the necessary conditions in terms of node activation and packet loss frequencies, when a particular communication protocol is adopted, are given in the next section. For simplicity, the convergence is formally shown fixing a communication protocol, but this is not restricting.

Remark 6.4.1. Like the Resilient Block Jacobi algorithm presented in Chapter 4, also the memory need of the robust asynchronous Newton-Raphson Consensus can be quite demanding. In fact, from one iteration to the following one, each agent has to store $5 + |\mathcal{N}_i^{\text{in}}|$ vectors of dimension n and $4 + |\mathcal{N}_i^{\text{in}}|$ matrices of dimension $n \times n$. If the feature space dimension n is large, it is possible to reduce the transmission and computational burden of matrix inversion. In particular, similarly to what has been proposed in Varagnolo et al. (2016), it is possible to modify the proposed algorithm to use Jacobi or Gradient descents which have reduced communication and computational requirements. More specifically, the only modification needed is to substitute line 30 with the following ones

$$\begin{aligned} H_i &\leftarrow \text{diag}(\nabla^2 f_i(\boldsymbol{x}_i)), && \text{Jacobi Descent Consensus,} \\ H_i &\leftarrow I_n, && \text{Gradient Descent Consensus.} \end{aligned}$$

As so, for the Jacobi Descent Consensus it is necessary to invert n scalars and to transmit only the n diagonal elements, while for the Gradient consensus no packets needs to be

transmitted as far as the Hessian is concerned. Of course, the price to pay with these choices is a likely slower convergence rate.
Note that a similar technique was applied in Chapter 4 with the introduction of the *resilient gradient descent* (see Remark 4.5.1).

Remark 6.4.2. As already pointed out in Remark 6.3.1, due to the locality of the results,it is possible to guarantee that $Z_i(k) \succ cI_n$ for all times k. However simulations showed that in order to increase the basin of attraction and the robustness of the algorithm it is suitable to force $Z_i(k) \geq cI_n$. A simple solution is to replace line (27) with

$$\boldsymbol{x}_i \leftarrow (1-\epsilon)\boldsymbol{x}_i + \epsilon[Z_i]_c^{-1}\boldsymbol{y}_i$$

where the operator $[\cdot]_c$ is defined as

$$[Z]_c := \begin{cases} Z & \text{if } Z \succeq cI_n \\ cI_n & \text{otherwise.} \end{cases}$$

where $Z \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix. This does not impair the local stability analysis provided below since close to the equilibrium point we have $[Z_i(k)]_c^{-1} = (Z_i(k))^{-1}$.

6.5 Dynamical system interpretation of ra-NRC

In this section, Algorithm 6.1 is rewritten as a dynamical system (which allows to study the convergence properties of the algorithm). To do so, it is necessary to define the evolution of the flags $\texttt{flag}_{\texttt{update}}$, $\texttt{flag}_{\texttt{transmission}}$, $\texttt{flag}_{\texttt{reception}}$, which can be done choosing an asynchronous protocol for the communication and modeling the packet losses. As will be shown later, the choice of the communication protocol is not restricting. In order to keep the notation lighter, from now on, only to the scalar case is considered, i.e., $x_i \in \mathbb{R}$ for all i. Consistently, the first and second derivatives of the function f_i are denoted as f_i' and f_i'' respectively .

The following analysis is done for an *asymmetric broadcast* communication protocol subject to packet losses. At each time instant k one node, say i, is activated. Then, node i performs in order the operations in the *Estimate Update* block and in the *Data Transmission* block, broadcasting to all its out-neighbors in $\mathcal{G}$ the updated variables $\sigma_{i,y}, \sigma_{i,z}$. The transmitted packet might be received or not by $j \in \mathcal{N}_i^{\text{out}}$, depending whether the link (i,j) is reliable or not at the time of transmission. If (i,j) is reliable, then node j performs, in order, the operations in the *Data Reception* block, and in the *Estimate Update* block.

An algorithmic description of the *asymmetric broadcast* communication protocol with packet losses for the ra-NRC Algorithm 6.1 is provided in Algorithm 6.2. Without loss of generality, the node performing the transmission step during the k-th iteration is node i.

Algorithm 6.2 Asymmetric broadcast for ra-NRC algorithm

Node i is activated

1: $\texttt{flag}_{\texttt{update},i} \leftarrow 1$ (line 26) : *Estimate Update*
2: $\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$ (line 8) : *Data transmission*

For $j \in \mathcal{N}_i^{\text{out}}$, if (i,j) is reliable

3: $\texttt{flag}_{\texttt{reception},j} \leftarrow 1$ (line 17) : *Data reception*
4: $\texttt{flag}_{\texttt{update},j} \leftarrow 1$ (line 26) : *Estimate Update*

The protocol selected allows to rewrite the resulting ra-NRC as a dynamical system of the form:

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon\phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)), \end{cases}$$

where proper definitions of variables $\boldsymbol{x}$, $\boldsymbol{\xi}$ and maps ϕ and φ can be found in Corollary 6.5.2.

Next, for the sake of analysis, a sequential description of the ra-NRC algorithm obtained adopting the communication protocol in Algorithm 6.2, is given. Observe that, once activated, node i updates x_i, g_i^{old}, h_i^{old}, g_i, h_i according to lines 27, 28, 29, 30, 31, i.e.,[4]

$$\begin{aligned} x_i(k+1) &= (1-\epsilon)x_i(k) + \epsilon\,[z_i(k)]_c^{-1}\, y_i(k) \\ g_i^{\text{old}}(k+1) &= g_i(k) \\ h_i^{\text{old}}(k+1) &= h_i(k) \\ g_i(k+1) &= f_i''(x_i(k+1))x_i(k+1) - f_i'(x_i(k+1)) \\ h_i(k+1) &= f_i''(x_i(k+1)). \end{aligned}$$

Based on $g_i(k+1)$ and $h_i(k+1)$, the variables y_i and z_i are updated performing in order the steps in lines 32, 10, and 33, 11, respectively, which result in

$$\begin{aligned} y_i(k+1) &= \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1)\right) \\ z_i(k+1) &= \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(z_i(k) + h_i(k+1) - h_i^{\text{old}}(k+1)\right), \end{aligned}$$

[4]Since $f_i \colon \mathbb{R} \mapsto \mathbb{R}$, all the quantities are now scalars.

and, in turn, from lines 12, 13,

$$\sigma_{i,y}(k+1) = \sigma_{i,y}(k) + y_i(k+1)$$
$$\sigma_{i,z}(k+1) = \sigma_{i,z}(k) + z_i(k+1).$$

The quantities $\sigma_{i,y}(k+1)$, $\sigma_{i,z}(k+1)$ are transmitted by node i to its out-neighbors; if (i,j) is reliable, then node j, based on the *Data Reception* packet, updates the local variables y_j, z_j, $\rho_{j,y}^{(i)}$, $\rho_{j,z}^{(i)}$ as[5]

$$\begin{aligned} y_j' &= y_j(k) + \sigma_{i,y}(k+1) - \rho_{j,y}^{(i)}(k) \\ z_j' &= z_j(k) + \sigma_{i,z}(k+1) - \rho_{j,z}^{(i)}(k) \\ \rho_{j,y}^{(i)}(k+1) &= \sigma_{i,y}(k+1) \\ \rho_{j,z}^{(i)}(k+1) &= \sigma_{i,z}(k+1) \end{aligned}$$

and, subsequently, based on the *Data Update* packet, updates the local variables x_j, g_j^{old}, h_j^{old}, g_j, h_j, y_j, z_j as

$$\begin{aligned} x_j(k+1) &= (1-\epsilon)x_j(k) + \epsilon \frac{y_j(k)}{[z_j(k)]_c} \\ g_j^{\text{old}}(k+1) &= g_j(k) \\ h_j^{\text{old}}(k+1) &= h_j(k) \\ g_j(k+1) &= f_j''(x_j(k+1))x_j(k+1) - f_j'(x_j(k+1)) \\ h_j(k+1) &= f_j''(x_j(k+1)) \\ y_j(k+1) &= y_j' + g_j(k+1) - g_j^{\text{old}}(k+1) \\ z_j(k+1) &= z_j' + h_j(k+1) - h_j^{\text{old}}(k+1). \end{aligned}$$

Next, a suitable vector-form description of the asymmetric broadcast ra-NRC algorithm is given.

To do so, similarly to the ra-AC algorithm presented in Chapter 5, it is first necessary to build an **augmented** network that contains all the nodes in V and also some additional virtual nodes. In particular, for each $(i,j) \in \mathcal{E}$ a new node, denoted as (i,j) for convenience, is introduced, and is connected to the remaining network as an out-neighbor of node i and an in-neighbor of node j. Formally, denoting the augmented network by

[5] As far as the variables y_j and z_j are concerned, to denote their updates in the *Data Reception packet* we introduce the auxiliary variables y_j', z_j', since the overall updates of the current values of y_j and z_j are performed in the subsequent *Data Update* packet.

$\mathcal{G}_a = (V_a, \mathcal{E}_a)$, it holds $V_a = V \cup \mathcal{E}$ and

$$\mathcal{E}_a = \mathcal{E} \cup \{(i,(i,j)) \mid (i,j) \in \mathcal{E}\} \cup \{((i,j),j) \mid (i,j) \in \mathcal{E}\}.$$

Similarly to what was done in Section 5.10, the auxiliary variables for each $(i,j) \in \mathcal{E}$ are $\nu_{j,y}^{(i)}(k)$, $\nu_{j,z}^{(i)}(k)$, defined as

$$\begin{aligned}\nu_{j,y}^{(i)}(k) &= \sigma_{i,y}(k) - \rho_{j,y}^{(i)}(k)\\ \nu_{j,z}^{(i)}(k) &= \sigma_{i,z}(k) - \rho_{j,z}^{(i)}(k).\end{aligned}$$

Recall that the role of the above variables is to keep track of the transmitted mass, which has not been received due to packet losses. Accordingly, let $\boldsymbol{\nu}_y$ and $\boldsymbol{\nu}_z$ be the vectors that collect, respectively, all the variables $\nu_{j,y}^{(i)}$ and $\nu_{j,z}^{(i)}$, $i \in V$ and $j \in \mathcal{N}_i^{\text{out}}$. Assuming that $|\mathcal{E}| = N_{\mathcal{E}}$, then $\boldsymbol{\nu}_y$, $\boldsymbol{\nu}_z \in \mathbb{R}^{N_{\mathcal{E}}}$. Now, define vectors $\boldsymbol{y}$, $\boldsymbol{z} \in \mathbb{R}^N$ as

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \boldsymbol{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_N \end{bmatrix},$$

and, based on these vectors, build the augmented vectors $\boldsymbol{y}_a$, $\boldsymbol{z}_a \in \mathbb{R}^{N+N_{\mathcal{E}}}$ as

$$\boldsymbol{y}_a = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\nu}_y \end{bmatrix}, \qquad \boldsymbol{z}_a = \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\nu}_z \end{bmatrix}.$$

Moreover, let

$$\begin{aligned}\boldsymbol{g} &= [g_1, \dots, g_N]^T,\\ \boldsymbol{g}^{\text{old}} &= \left[g_1^{\text{old}}, \dots, g_N^{\text{old}}\right]^T,\\ \boldsymbol{f}''(\boldsymbol{x})\boldsymbol{x} &= [f_1''(x_1)\, x_1, \dots, f_N''(x_N)\, x_N]^T,\\ \boldsymbol{f}'(\boldsymbol{x}) &= [f_1'(x_1), \dots, f_N'(x_N)]^T,\\ \boldsymbol{y}/\boldsymbol{z} &= \left[\frac{y_1}{z_1}, \dots, \frac{y_N}{z_N}\right]^T.\end{aligned}$$

Since the communication scenario is lossy, it might happen that the packet transmitted by node i is either received or not received by node $j \in \mathcal{N}_i^{\text{out}}$. For this reason, it is convenient to introduce the sets

$$\tilde{\mathcal{N}}_i(k) = \left\{ j \in \mathcal{N}_i^{\text{out}} \text{ such that } (i,j) \text{ is reliable at time } k \right\},$$

and, its complement on $\mathcal{N}_i^{\text{out}}$,

$$\bar{\mathcal{N}}_i(k) = \mathcal{N}_i^{\text{out}} \setminus \tilde{\mathcal{N}}_i(k).$$

To state Proposition 6.5.1, which provides a vector form description of Algorithm 6.2, it is convenient to resort to the following notational convention. When referring to an N-dimensional vector, its components are indexed according to the nodes in V, while when referring to an $N_{\mathcal{E}}$-dimensional vector, its components are indexed according to the edges in $\mathcal{E}$. In particular, $e_i \in \mathbb{R}^N$ and $e_{(i,j)} \in \mathbb{R}^{N_{\mathcal{E}}}$ denote the vectors with all the components equal to zero, except, respectively, the one related to node i and the one related to edge (i,j), which are equal to one; that is e_i, $i \in V$, and $e_{(i,j)}$, $(i,j) \in \mathcal{E}$, are the vectors of the canonical basis of, respectively, $\mathbb{R}^N$ and $R^{N_{\mathcal{E}}}$.

Proposition 6.5.1. *The ra-NRC algorithm with the asymmetric broadcast protocol (Algorithm 6.1 and Algorithm 6.2), can be written in vector form as*[6]

$$\begin{aligned}
\boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \epsilon\, S(k)\ (\boldsymbol{p}(k) - \boldsymbol{x}(k)) \\
\boldsymbol{g}^{old}(k+1) &= S(k)\boldsymbol{g}(k) + (I - S(k))\boldsymbol{g}^{old}(k) \\
\boldsymbol{g}(k+1) &= \boldsymbol{f}''(\boldsymbol{x}(k+1))\boldsymbol{x}(k+1) - \boldsymbol{f}'(\boldsymbol{x}(k+1)) \\
\boldsymbol{h}^{old}(k+1) &= S(k)\boldsymbol{h}(k) + (I - S(k))\boldsymbol{h}^{old}(k) \\
\boldsymbol{h}(k+1) &= \boldsymbol{f}''(\boldsymbol{x}(k+1)) \\
\boldsymbol{y}_a(k+1) &= M(k)\boldsymbol{y}_a(k) + T(k)\ \Big(\boldsymbol{g}(k+1) - \boldsymbol{g}^{old}(k+1)\Big) \\
\boldsymbol{z}_a(k+1) &= M(k)\boldsymbol{z}_a(k) + T(k)\ \Big(\boldsymbol{h}(k+1) - \boldsymbol{h}^{old}(k+1)\Big) \\
\boldsymbol{p}(k+1) &= \frac{\boldsymbol{y}(k+1)}{\boldsymbol{z}(k+1)}
\end{aligned} \tag{6.7}$$

where

$$S(k) = e_i e_i^T + \sum_{j \in \tilde{\mathcal{N}}_i(k)} e_j e_j^T \quad \textit{and} \quad T(k) = \begin{bmatrix} T_V(k) \\ T_{\mathcal{E}}(k) \end{bmatrix},$$

[6] Matrices S, S_a and M depend on which node is activated, and on which edges between this node and its out-neighbors are reliable. In order to keep the notation lighter, this dependency is not made explicit (for instance using some superscript or subscript); instead, only the time-varying nature of these matrices is underlined, writing $S(k)$, $S_a(k)$ and $M(k)$.

with

$$T_V(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\left(e_i e_i^T + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j e_i^T\right) + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j e_j^T$$
$$T_{\mathcal{E}}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_i^T$$

and where $M(k)$ is a column stochastic matrix such that

$$M(k) = \begin{bmatrix} M_{VV}(k) & M_{V\mathcal{E}}(k) \\ M_{\mathcal{E}V}(k) & M_{\mathcal{E}\mathcal{E}}(k) \end{bmatrix}$$

with

$$M_{VV}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\left(e_i e_i^T + \sum_{j\in\tilde{\mathcal{N}}_i} e_j e_i^T\right) + \sum_{h\neq i} e_h e_h^T$$
$$M_{V\mathcal{E}}(k) = \sum_{j\in\tilde{\mathcal{N}}_i} e_j e_{(i,j)}^T$$
$$M_{\mathcal{E}V}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_i^T \quad (= T_{\mathcal{E}}(k))$$
$$M_{\mathcal{E}\mathcal{E}}(k) = \sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_{(i,j)}^T + \sum_{(r,s):\, r\neq i} e_{(r,s)} e_{(r,s)}^T.$$

The proof of the latter proposition can be found in Appendix E.1.

Observe that variables $\boldsymbol{y}_a, \boldsymbol{z}_a$ are trajectories of a linear, time-varying algorithm with column-stochastic state-matrix, driven by the differences $\boldsymbol{g} - \boldsymbol{g}^{\text{old}}$, $\boldsymbol{h} - \boldsymbol{h}^{\text{old}}$.

From the previous proposition, the next fact follows directly.

Corollary 6.5.2. *Let $\boldsymbol{\xi} = \left[\boldsymbol{g}^T, \boldsymbol{g}^{oldT}, \boldsymbol{h}^T, \boldsymbol{h}^{oldT}, \boldsymbol{y}_a^T, \boldsymbol{z}_a^T, \boldsymbol{p}^T\right]^T$, then, system in (6.7) can be written as:*

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon\, \phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)), \end{cases} \tag{6.8}$$

where $\epsilon > 0$, $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{\xi} \in \mathbb{R}^{7N+2N_{\mathcal{E}}}$, $\phi : \mathbb{N} \times \mathbb{R}^N \times \mathbb{R}^{7N+2N_{\mathcal{E}}} \to \mathbb{R}^N$, $\varphi : \mathbb{N} \times \mathbb{R}^N \times \mathbb{R}^{7N+2N_{\mathcal{E}}} \to \mathbb{R}^{7N+2N_{\mathcal{E}}}$, and where equations in (6.7) properly define the maps ϕ and φ.

Finally, the *mass conservation* property of system in (6.7), stated in the following lemma, will be useful in the next section.

Lemma 6.5.3. *Consider system in (6.7). Then, for all $k \in \mathbb{N}$, the following equalities hold true*

$$\sum_{\ell=1}^{N} \left(y_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{out}} \nu_{j,y}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} g_\ell(k)$$
$$\sum_{\ell=1}^{N} \left(z_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{out}} \nu_{j,z}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} h_\ell(k).$$

The proof of this lemma can be found in Appendix E.2.

Remark 6.5.4. The dynamical system description of ra-NRC algorithm given in this section has been obtained using an asymmetric broadcast communication protocol. However, it is worth stressing that similar computations hold also for other communication protocols like *symmetric gossip*, *asymmetric gossip*, *coordinated broadcast*. Adopting one of the above communication protocols, it turns out that ra-NRC algorithm can again be described as in (6.7), with the only difference related to the matrix $M(k)$ which is still a column stochastic matrix but with a slight different structure, and to the selection matrix $S(k)$. This justifies the fact that the convergence results provided in the next Section, specifically tailored to the scenario considered in this section, can be technically extended to also other types of communication protocols.

6.6 Theoretical analysis of the ra-NRC

This section is devoted to the theoretical analysis of the asymmetric broadcast ra-NRC algorithm, presented in the previous section.

As for all the other algorithms developed in the thesis, to prove the convergence of the asymmetric broadcast ra-NRC some assumptions concerning the node activation and on the packet losses are needed. As stands to reason, the assumptions needed are the same one used in the previous chapter to show the convergence of ra-AC, that is Assumptions 2.3.2 and 2.3.1. Namely, each node updates its local variables and communicates with its neighbors infinitely often, and the number of consecutive packet losses is bounded. As for the previous chapters, from the two assumptions it follows that, given $i \in V$ and $j \in \mathcal{N}_i^{\text{out}}$, node j receives information from node i at least once within the interval $[k, k + h\tau]$.

In order to characterize the convergence properties of the asymmetric broadcast ra-NRC algorithm, two lemmas are introduced.

Let $\boldsymbol{x} = [x_1, \dots, x_N]^T$ and $\boldsymbol{x}^0 = [x_1^0, \dots, x_N^0]^T$. The first lemma shows that if the variable $\boldsymbol{x}$ is kept constant, then the components of the vector $\boldsymbol{p}$ achieve consensus to the ratio $\overline{h}(x_1, \dots, x_N)/\overline{g}(x_1, \dots, x_N)$. Vice-versa, the second lemma, shows that if the components of $\boldsymbol{p}$ have reached consensus, then the vector $\boldsymbol{x}$ exponentially converges to the global minimizer.

Formally, to state the first result, for a given $\bar{k}$, it is useful to consider the following dynamics, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}_{\bar{k}}(k+1) = \varphi\left(k, \boldsymbol{x}(\bar{k}), \boldsymbol{\xi}_{\bar{k}}(k)\right), \tag{6.9}$$

initialized by $\boldsymbol{\xi}_{\bar{k}}(\bar{k}) = \boldsymbol{\xi}(\bar{k})$. Observe that, $\boldsymbol{\xi}_{\bar{k}}$ describes the evolution of the variable $\boldsymbol{\xi}$, starting at iteration $\bar{k}$, assuming that the variable $\boldsymbol{x}$ is kept constant for $k \geq \bar{k}$, that is, $\boldsymbol{x}(k) = \boldsymbol{x}(\bar{k})$ for all $k \geq \bar{k}$. In particular, the interest is on the behavior of the variable $\boldsymbol{p}$, the last block of components of $\boldsymbol{\xi}_{\bar{k}}$. Similarly to $\boldsymbol{\xi}_{\bar{k}}$, $\boldsymbol{p}$ in the given scenario is denoted as $\boldsymbol{p}_{\bar{k}}$. The following result holds:

Lemma 6.6.1. *For a given $\bar{k}$, consider, for $k \geq \bar{k}$, the dynamics in (6.9). Then, under Assumptions 2.3.2 and 2.3.1, the point*

$$\frac{\sum_\ell g_\ell(x_\ell(\bar{k}))}{\sum_\ell h_\ell(x_\ell(\bar{k}))}\mathbf{1}$$

is exponentially stable for the variable $\boldsymbol{p}_{\bar{k}}$, that is, defined

$$\tilde{\boldsymbol{p}}_{\bar{k}}(k) := \boldsymbol{p}_{\bar{k}}(k) - \frac{\sum_\ell g_\ell(x_\ell(\bar{k}))}{\sum_\ell h_\ell(x_\ell(\bar{k}))}\mathbf{1},$$

there exists $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$ such that

$$\|\tilde{\boldsymbol{p}}_{\bar{k}}(\,k\,)\| \leq C_{\bar{k}}\, \rho_{\bar{k}}^{k-\bar{k}}\, \|\tilde{\boldsymbol{p}}_{\bar{k}}(\,\bar{k}\,)\|. \tag{6.10}$$

Proof. In the following the block components of $\boldsymbol{\xi}_{\bar{k}}$ corresponding to $\boldsymbol{y}_a$, $\boldsymbol{z}_a$ are denoted by $\boldsymbol{y}_{a;\bar{k}}$, $\boldsymbol{z}_{a;\bar{k}}$. To study the evolution of $\boldsymbol{p}_{\bar{k}}(k)$, the behavior of the variables $\boldsymbol{y}_{a;\bar{k}}(k)$ and $\boldsymbol{z}_{a;\bar{k}}(k)$ are analyzed separately. Consider $\boldsymbol{y}_{a;\bar{k}}(k)$. Observe that, since $\boldsymbol{x}(k) = \boldsymbol{x}(\bar{k})$, $k \geq \bar{k}$, according to Assumptions 2.3.2 and 2.3.1 there exists $\bar{k}' > \bar{k}$ such that $\boldsymbol{g}^{\text{old}}(k) = \boldsymbol{g}(k)$ for all $k \geq \bar{k}'$ and, hence,

$$\boldsymbol{y}_{a,\bar{k}}(k+1) = M(k)\boldsymbol{y}_{a,\bar{k}}(k),$$

for $k \geq \bar{k}'$. A similar reasoning holds for $\boldsymbol{z}_{a,\bar{k}}(k)$. It follows that the variables $\boldsymbol{y}_{a,\bar{k}}(k)$, $\boldsymbol{z}_{a,\bar{k}}(k)$ and, in turn, the variables $\boldsymbol{y}_{\bar{k}}(k)$, $\boldsymbol{z}_{\bar{k}}(k)$ for $k \geq \bar{k}'$ run the same iterations of the variable $\boldsymbol{q}_a(k)$ in the ra-AC algorithm introduced in the previous chapter.

From Lemma 6.5.3, for $k \geq \bar{k}$ and, in particular, for $k \geq \bar{k}'$

$$\mathbf{1}^T \boldsymbol{y}_{a,\bar{k}}(k) = \sum_{\ell=1}^{N} g_\ell\left(x_\ell(\bar{k})\right)$$
$$\mathbf{1}^T \boldsymbol{z}_{a,\bar{k}}(k) = \sum_{\ell=1}^{N} h_\ell\left(x_\ell(\bar{k})\right).$$

From Theorem 5.9.4, it follows that $\frac{\boldsymbol{y}_{\bar{k}}(k)}{\boldsymbol{z}_{\bar{k}}(k)}$ converges exponentially to $\frac{\sum_\ell g_\ell(x_\ell(\bar{k}))}{\sum_\ell h_\ell(x_\ell(\bar{k}))}\mathbf{1}$. ■

Now, assume that, for each k, the variable $\boldsymbol{p}(k)$ has reached consensus and consider the following dynamics for the variable $\boldsymbol{x}$,

$$\begin{aligned}\boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \epsilon\, S(k)\left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right)\\ &= \boldsymbol{x}(k) + \epsilon\,\tilde{\phi}(k;\boldsymbol{x}(k))\end{aligned} \tag{6.11}$$

where

$$\tilde{\phi}(k;\boldsymbol{x}(k)) = S(k)\left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right). \tag{6.12}$$

Let

$$\boldsymbol{x}^* = x^*\mathbf{1}, \tag{6.13}$$

and recall that x^* is the minimizer of the optimization problem in (6.1). One can see that $\boldsymbol{x}^*$ is an equilibrium of (6.11) by simply plugging each component $\boldsymbol{x}^*_\ell = x^*$ of $\boldsymbol{x}^*$ into (6.12) and noting that, from the optimality of x^*, $\sum_\ell g_\ell(x^*) = \sum_l [f_l''(x^*)x^* - f_l'(x^*)] = \sum_l f_l''(x^*)x^*$ and $\sum_l h_l(x^*) = \sum_l f_l''(x^*)$. The following result states that the linearized version of (6.11) around $\boldsymbol{x}^*$ is an exponentially stable system.

Lemma 6.6.2. *Consider system in* (6.11) *and let* $\boldsymbol{x}^*$ *be as in* (6.13). *Let*

$$A(k) = I + \epsilon \frac{\partial \tilde{\phi}}{\partial \boldsymbol{x}}(k;\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*},$$

and, accordingly, consider the auxiliary system

$$\tilde{\boldsymbol{x}}(k+1) = A(k)\tilde{\boldsymbol{x}}(k). \tag{6.14}$$

Then, under Assumptions 2.3.2 and 2.3.1, $\tilde{\boldsymbol{x}} = 0$ *is an exponentially stable equilibrium point for* (6.14).

Proof. Let

$$\alpha(\boldsymbol{x}(k)) = \frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}.$$

Computing the partial derivative of α with respect to x_i, it holds

$$\left[\frac{\partial \alpha}{\partial x_i}\right]|_{\boldsymbol{x}=\boldsymbol{x}^*} = \frac{g_i'(x^*)\,\sum_{\ell=1}^N h_\ell\,(x^*) - h_i'\,(x^*)\,\sum_{\ell=1}^N g_\ell\,(x^*)}{\left(\sum_{\ell=1}^N h_\ell\,(x^*)\right)^2}$$

with

$$\begin{aligned}
&g_i'(x^*)\sum_{\ell=1}^N h_\ell\,(x^*) \;-\; h_i'\,(x^*)\sum_{\ell=1}^N g_\ell\,(x^*) = \\
&= (\,f_i'''(x^*)\,x^* \,+ f_i''(x^*) \;-\; f_i''(x^*)\,)\sum_{\ell=1}^N f_\ell''\,(x^*) - f_i'''(x^*)\sum_{\ell=1}^N (f_\ell''\,(x^*)\;x^* \;- f_\ell'(x^*)) \\
&= f_i'''(x^*)\,x^*\sum_{\ell=1}^N f_\ell''\,(x^*) \;-\; f_i'''(x^*)\,x^*\sum_{\ell=1}^N f_\ell''\,(x^*) + f_i'''(x^*)\sum_{\ell=1}^N f_\ell'(x^*) \\
&= 0,
\end{aligned}$$

where, in the last equality, the fact that $\sum_{\ell=1}^N f_\ell'(x^*) = 0$ was used. From the previous calculations, it turns out that

$$A(k) = I - \epsilon S(k).$$

By Assumption 2.3.2, matrix

$$\bar{A}_{k,\tau} = \prod_{s=k}^{k+\tau} A(k),$$

is a diagonal matrix such that $0 < \left[\bar{A}_{k,\tau}\right]_{ii} < 1 - \epsilon$, for all i. Then, system in (6.11) satisfies the stated property. ∎

Intuitively, one would conclude that, when the parameter ϵ is small, the results of the two lemma can be combined to simultaneously obtain asymptotic consensus and convergence to the global minimizer. This is formally shown in the next theorem which characterizes the convergence properties of the asymmetric broadcast ra-NRC algorithm.

Theorem 6.6.3. *Under Assumptions 2.3.2 and 2.3.1, and the assumptions posed in Section 6.2, there exist some positive scalars ϵ_c and δ such that, if the initial conditions $\boldsymbol{x}^o \in \mathbb{R}^N$ satisfy $\|\boldsymbol{x}^o - x^*\mathbf{1}\| < \delta$ and if ϵ satisfies $0 < \epsilon < \epsilon_c$ then the local variables x_i in Algorithm 6.1 are exponentially stable with respect to the global minimizer x^*.*

Proof. The proof of the result is based on showing that the system in (6.8) satisfies the

assumptions of Proposition E.3.2 in the Appendix. To do so, define, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k) = \tilde{I}\,\boldsymbol{\xi}_{\bar{k}}(k) \;+\; \tilde{u}, \tag{6.15}$$

where $\boldsymbol{\xi}_{\bar{k}}(k)$ is defined as in (6.9) and where

$$\tilde{I} = \begin{bmatrix} I_{(6N+2N_{\mathcal{E}})\times(6N+2N_{\mathcal{E}})} & 0_{(6N+2N_{\mathcal{E}})\times N} \\ 0_{N\times(6N+2N_{\mathcal{E}})} & 0_{N\times N} \end{bmatrix}$$

and

$$\tilde{u} = \begin{bmatrix} 0_{(6N+2N_{\mathcal{E}}\times 1} \\ \boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})} \end{bmatrix},$$

with

$$\boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})} = \frac{\sum_\ell g_\ell(x_\ell(\bar{k}))}{\sum_\ell h_\ell(x_\ell(\bar{k}))}\mathbf{1}.$$

Observe that the first six blocks components of $\boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k)$ coincide with the first six blocks components of $\boldsymbol{\xi}_{\bar{k}}(k)$, while the last block component is constant for all $k \geq \bar{k}$. Moreover, for $k \geq \bar{k}$, let

$$\tilde{\boldsymbol{\xi}}_{\bar{k}}(k) := \boldsymbol{\xi}_{\bar{k}}\,(k) - \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}\,(k)\,.$$

Based on the previous observation, the first six blocks components of $\tilde{\boldsymbol{\xi}}_{\bar{k}}(k)$ are equal to zero, while the last block component is equal to

$$\boldsymbol{p}_{\bar{k}}(k) - \boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})}.$$

Thus, $||\tilde{\xi}_{\bar{k}}(k)|| = ||\boldsymbol{p}_{\bar{k}}(k) - \boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})}||$, so that from Lemma 6.6.1, it follows that there exists $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$ such that

$$\|\tilde{\boldsymbol{\xi}}_{\bar{k}}(\,k\,)\| \leq\; C_{\bar{k}}\,\rho_{\bar{k}}^{k-\bar{k}}\;\|\tilde{\boldsymbol{\xi}}_{\bar{k}}(\,\bar{k}\,)\|. \tag{6.16}$$

This shows that system in (6.8) satisfies property in (E.8), in Appendix E.3

Consider now the system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \epsilon\,\phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right) && (6.17)\\ &= \boldsymbol{x}(k) + \epsilon\,S(k)\,\left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right) \\ &= \boldsymbol{x}(k) + \epsilon\,\tilde{\phi}(k;\boldsymbol{x}(k)) && (6.18) \end{aligned}$$

In Lemma 6.6.2, it is established that the previous system satisfies Assumption E.3.1,

in Appendix E.3. Hence, Proposition E.3.2, in Appendix E.3 can be applied to system in (6.8), yielding the result of the statement. ■

The major challenges in proving the main results are related to proving that the ra-NRC algorithm satisfy a number of technical conditions required by standard theory of separation of time-scales. Different conditions and theorems are available for continuous time dynamical systems (we refer the interested reader to Chapter 11 in Khalil (2001)). In particular, it is necessary to prove the exponential stability for a non-autonomous discrete time dynamical system whose closest counterpart in the continuous time is given by Theorem 11.4 in Khalil (2001).

Besides some standard conditions on smoothness and uniformity of the dynamical flows involved, there are three major requirements that need to be satisfied: the first is that the fast dynamics converges exponentially to an equilibrium manifold, the second is that the slow dynamics restricted to this manifold is exponentially stable, and the third is that a number of *bounded interconnection conditions* which represent the perturbation of the slow dynamics into the fast dynamics and vice-versa, are satisfied. As for the first requirement, the algorithm used for the consensus step, the ra-AC algorithm, is exponentially convergent (note that the work by Vaidya et al. (2011), on which ra-AC is based, only provides convergence in probability). As for the second one, proving the local exponential stability of the slow dynamics is not trivial since the dynamics is non-autonomous. As for the last requirement on the *bounded interconnection conditions*, very much depends on cost functions and in the discrete-time domain it is difficult to provide *global* guarantees. However, under some mild smoothness conditions, it is possible to show that the conditions on *bounded interconnection conditions* are locally satisfied, and, in turn, to prove local exponential stability.

Remark 6.6.4. Algorithm 6.1 assumes the initial conditions of the local variable x_i to be all identical to x^o. Although not being a very stringent requirement, this assumption can be relaxed, that is, slightly modified versions of Theorem 6.6.3 would hold even in the case x_i is initialized to x_i^o, as soon as all the initial conditions are sufficiently close to the global minimizer x^*, i.e., as soon as $|x_i^o - x^*| < \delta$ for all $i = 1, \dots, N$.
The initial conditions on the local variables $y_i = g_i^{\text{old}} = g_i = f_i''(x^o)x^o - f_i'(x^o)$ and $z_i = h_i^{\text{old}} = h_i = f_i''(x^o)$ are instead more critical for the convergence of the local variables x_i to the true minimizer x^*. As shown in Zanella et al. (2011), small perturbations of these initial conditions can lead to convergence to a point $\overline{x} \neq x^*$ (notice that these perturbations do not affect the stability of the algorithm, so that possible small numerical errors due to the computation and data quantization do not disrupt the convergence properties of the algorithm). Moreover, the map from the amplitude of these perturbations

and the distance $\|\overline{x} - x^*\|$ is continuous, so that if these perturbations are small then $\overline{x} \approx x^*$.

Remark 6.6.5. Although the previous theorem guarantees only local exponential convergence, numerical simulations on real datasets seem to indicate that the basin of attraction is rather large and stability is mostly dictated by the choice of the parameter ϵ. However, for the special but relevant case when the cost functions $f_i(x)$ are quadratic, as in distributed least-squares problems, local stability implies global stability Carli et al. (2015).

6.7 Numerical experiments

The problem analyzed is a regression one inspired by the UCI Housing dataset available at `http://archive.ics.uci.edu/ml/datasets/Housing`. In this task, an example $\boldsymbol{\chi}_j \in \mathbb{R}^{n-1}$ is a vector representing some features of a house (e.g., per-capita crime rate by town, index of accessibility to radial highways, etc.), and $y_j \in \mathbb{R}$ denotes the corresponding median monetary value of of the house. The objective is to obtain a predictor of house value based on these data. Due to privacy issues, the data are divided among $N = 10$ nodes. These nodes can communicate according to a random geometric graph with distance threshold $r = 0.5$. The regression problem can be formulated as a distributed problem defined on the local costs

$$f_i(\boldsymbol{x}) := \sum_{j \in \mathcal{F}_i} \frac{\left(y_j - \boldsymbol{\chi}_j^T \boldsymbol{x}' - x_0\right)^2}{\left|y_j - \boldsymbol{\chi}_j^T \boldsymbol{x}' - x_0\right| + \beta} + \gamma \left\|\boldsymbol{x}'\right\|_2^2. \tag{6.19}$$

where $\boldsymbol{x} = (\boldsymbol{x}', x_0) \in \mathbb{R}^{n-1} \times \mathbb{R}$ is the vector of coefficient for the linear predictor $\widehat{y} = \boldsymbol{\chi}^T \boldsymbol{x}' + x_0$ and γ is a common regularization parameter. The loss function $\frac{(\cdot)^2}{|\cdot|+\beta}$ corresponds to a smooth version of the Huber robust loss, a loss that is usually employed to minimize the effects of outliers. In this case β dictates for which arguments the loss is pseudo-linear or pseudo-quadratic and has been manually chosen to minimize the effects of outliers. In the experiments the following parameters are used: $n = 3$ number of features, $\beta = 50$, $\gamma = 1$, and $|\mathcal{F}| = 506$ total number of examples in the dataset. The examples are randomly assigned to the $N = 10$ agents. The performance index considered is the Mean Squared Error

$$MSE(k) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i(k) - \boldsymbol{x}^*\|.$$

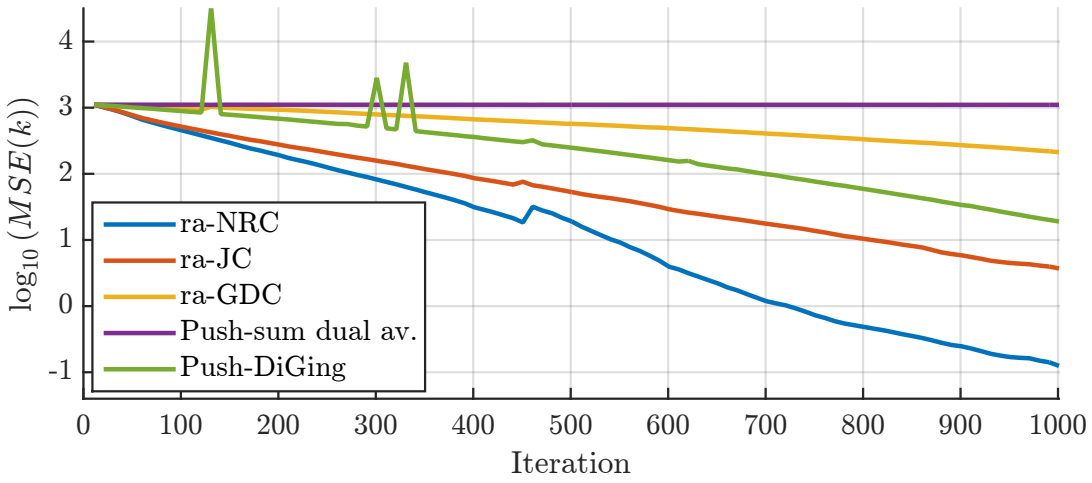


Figure 6.2: Evolution of the MSE for different optimization algorithms. Concerning the choice of the hyperparamenters, ϵ for ra-NRC is 0.04, ϵ for ra-JC is 0.04, ϵ for ra-GDC is 0.0008, α for Push-DIGing is 0.0006 and α for Push-sum distributed dual averaging is 0.002.

The first experiment concerns a lossless communication scenario, and compare the convergence rate of five schemes: the ra-NRC algorithm 6.1 and its Jacobi and Gradient Descent versions defined in Remark 6.4.1, plus the Push-DIGing and Push-sum distributed dual averaging algorithms defined respectively in Nedić et al. (2016) and Tsianos et al. (2012). These two last schemes are based on broadcast-like communications but do not handle lossy communications.

Figure 6.2 plots the evolution of the MSEs for the different algorithms subject to the same communication sequence (i.e., at the same time instant k the same node i awakens). The hyperparameters defining each of the algorithms are individually tuned. In particular, a grid of 20 logarithmically spaced potential values between 10^{-4} and 1 have been tested for each algorithm. The hyperparameter selected for the plot is the one according to which the algorithm achieves the fastest convergence speed (the selection is therefore a posteriori).

Figure 6.3 instead inspects the effect of varying the probability of packets losses on the MSEs of single realizations for the ra-NRC scheme. In this figure the algorithm is tuned with a slower $\epsilon = 0.02$, to highlight the intuition that increasing the chances of packet losses leads to slower convergence properties.

The last simulations contain a comparison between ra-NRC and PDMM. The communication graph is random geometric with $N = 10$ nodes and with distance threshold equal to 0.3, and the cost function is again 6.19. Figure 6.4 shows that, at least for the cost functions used in these simulations, ra-NRC is competitive with PDMM (however, it

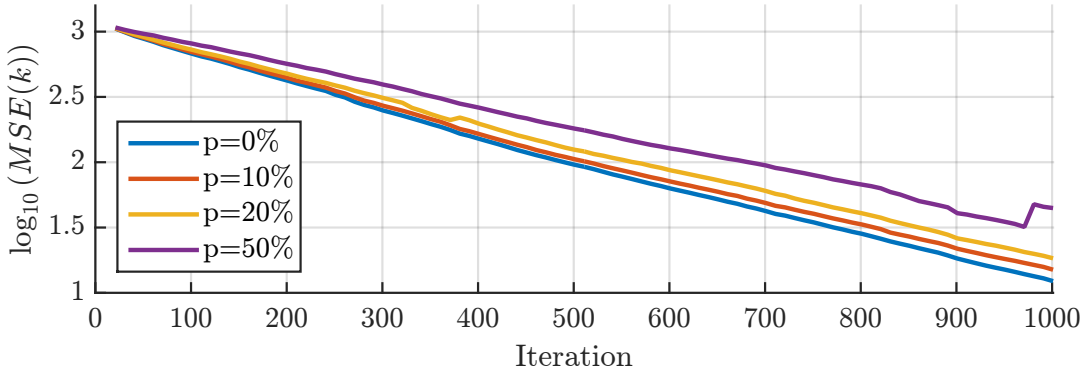


Figure 6.3: MSEs for the ra-NRC scheme for different values of packet loss probability p. The problem is the same one considered in Figure 6.2. The step size ϵ for the ra-NRC is set to 0.02 for all the different packet loss probabilities.

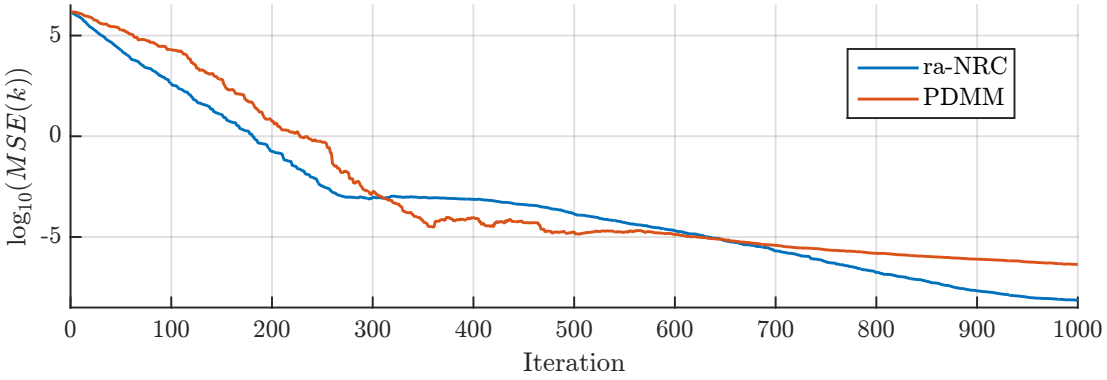


Figure 6.4: MSEs for the ra-NRC scheme and for PDMM. The step size ϵ for the ra-NRC is set to 0.08 and the value for α and ρ for PDMM have been chosen equal to 1 and 7, where the parameters are manually tuned (to the best of the author's possibility to give the best convergence for both).The packet loss probability is set to 20%.

would be necessary to perform comparison also on other cost functions types, which was not possible in this thesis due to time issues since PDMM was discovered really close to the dissertation's deadline). Some further simulations have been carried out to study the algorithms' sensitivity to the parameters' settings. These showed that, if the change is of the order of 10% or 20%, the performance of both algorithms does not change too much.

Remark 6.7.1. PDMM, as usual for Lagrangian based methods, has minimization steps to be solved inside each iteration. As so, the computational time depends on how efficiently this minimization steps can be carried out. In the simulations shown if Figure 6.4, the execution time to perform 1000 iteration is 17.18 seconds for the ra-NRC, while for PDMM is 97.22 seconds (the minimization step for each iteration of the PDMM is solved using function fminunc on MATLAB 2015b®).

6.8 Final considerations

This chapter addressed the problem of distributed unconstrained convex optimization in the context of lossy communication. This scenario is almost not treated in the literature, except for the notable work Sherson et al. (2017), which is the only work found by the author in the literature which is able to cope with a lossy communication scenario. More specifically, a robustified version of the Newton-Raphson consensus algorithm originally proposed in Zanella et al. (2011) was proposed. Its (local) convergence properties are proven under some general mild assumptions on the local costs and on the communication.In particular, the considered optimization strategy is locally exponentially stable around the global optimum as soon as the local costs are $\mathcal{C}^3$ and strongly convex with second derivative bounded from below. The simulations on real datasets compare favourably against some alternative first-order algorithms available in the literature even under the special case of lossless asynchronous scenario.

Possible future research directions include adaptive strategies to tune the step-size ϵ on-line, the inclusion of equality constraints of the form $Ax = b$, and the extension of partition-based approaches where each agent is interested in computing only some components of the global minimizer vector (i.e., to the kind of cost functions studied in Chapter 4).

7

Conclusions and future directions

This final chapter draws the conclusion of the entire dissertation. The common thread which connects all the chapters is certainly the strong emphasis given to to the communication step of a distributed algorithm. In particular, the thesis aimed at presenting algorithms which do not assume a reliable communication network.

In the literature, the majority of the works in distributed optimization assume reliable communication; moreover, they usually rely on a synchronous update and communication step. Note that, from a practical point of view, if no acknowledge system is implemented, almost all kind of communication is prone to packet losses. As so, when a developed distributed algorithm does not consider packet losses, the communication protocol employed by the agents has an acknowledge system. Recently, some of the literature in distributed optimization has focused on asynchronous updates, since they require less coordination among agents. Concerning lossy communication networks, however, the existing algorithms in the literature which can deal with such networks assume that the node sending information perfectly knows its out-neighbors set (in a way, a lossy communication network is considered as a time varying communication network). This implies again the presence of an acknowledge system or of a system which allows to test if the link between two agents is reliable at the transmission time. Even though a protocol with ACKs might not be difficult to implement, it can make the algorithm slower, and it requires more energy consumption (the latter being an issue only for some kind of

multi-agent systems with limited battery supplies). The only (to the best of the author's knowledge) notable exception to the use of ACKs in order to cope with an unreliable (and asynchronous) communication scenario, is PDMM Sherson et al. (2017). This very recent algorithm is used for comparison in Chapters 5 and 6.

The aim of the dissertation was to present some algorithms which can cope with an unreliable communication set-up without requiring an acknowledge system. The problems analyzed in the chapters are different. Accordingly, also the developed algorithms and convergence proof are different, even though some of the tools used are common.

Chapter 3 was devoted to the development of a robust distributed algorithm for the patrolling problem. The robust algorithm is obtained starting from a non-robust one through a quite simple modification. The proof of convergence of the robustified one, however, required more effort. The tools used to prove convergence concern Lyapunov theory and convergence of dynamical switching systems. The algorithm obtained is task-dependent; as a consequence, its applicability to other problems is limited.

The remaining chapters, on the other hand, deal with more general problems. In particular the minimization of the sum of cost functions.

In Chapter 4, the local cost functions are locally coupled. The focus is therefore on developing an algorithm which actually exploits this locality of the information, which also allows the agents in the network to estimate just their own part of the optimization variable. This, in turn, allows to better preserve privacy. In particular, to perform the update step, each agent has to require only information coming from one-hop neighbors. The "trick" employed to force communication only between direct neighbors, i.e. keeping in memory the last received packets, simultaneously allows to cope with an unreliable communication network. The algorithm proposed is based on the well-known Jacobi iteration, and the convergence proof exploits Lyapunov theory and separation of time scale principle.

The other two chapters, that is Chapters 5 and 6, are tightly connected. In this case, the problem to solve was the minimization of the sum of cost functions, where the local cost functions (possibly) depend on the whole optimization variable. In this case, the agents are interested in retrieving the whole solution (in other words, the problem is a global estimation one).

Chapter 5 dealt with the consensus/quadratic minimization problem. The first part of the chapter was devoted to the comparison between consensus based and Lagrangian based algorithms to solve the aforementioned problems. This comparison showed the importance of the communication graph in determining the convergence rate of distributed algorithms. Moreover, it also underlined the fact that the convergence rate of Lagrangian

based algorithms is sensitive to the curvature of the quadratic cost function. The second part of the chapter was more in line with the other chapters, since it was devoted to the design of a robust and asynchronous algorithm to solve the consensus problem, the ra-AC algorithm. This algorithm is based on the ratio-consensus one, which is made robust introducing additional variables that in a way summarize the past communication history. This algorithm was shown to be exponentially convergent using ergodicity theory for stochastic matrices.

Finally, Chapter 6 addressed the minimization of the sum of convex cost functions (not necessarily quadratic as in the previous scenario) in case of non-ideal communication. The algorithm introduced, ra-NRC, is a robustification of the Newton-Raphson Consensus Zanella et al. (2011), and is obtained merging the latter with the ra-AC algorithm developed in Chapter 5. The convergence proof relies on Lyapunov and time-scale separation theory. The proof is quite involved since a part of the state variable which describes the algorithm does not converge, even though the part regarding the estimate of the minimizer converges to the true minimizer.

There are many possible future research directions, some of them regarding open problems also in the literature.

On-line step-size adaptation Concerning the algorithms in Chapters 4 and 6, an important aspect to study more deeply is the selection of the step size ϵ. Its choice is of paramount importance, since a too big ϵ results in a diverging algorithm, and a too small one results in a very slow algorithm. It would be interesting to find an adaptive strategy to tune the step size ϵ while the algorithm is running; this should help to increase the rate of convergence of the algorithms. The adaptive step-size selection is an open problem for the majority of the optimization algorithms in the literature.

Constrained optimization Except for the first problem, which was constrained, all the other problems considered in this manuscript are unconstrained. As so, an important way to improve the applicability of the proposed solutions is to find a way to employ them to solve also constrained optimization problems, for example problems where the solution has to satisfy linear equality and inequalities constraints $Ax = b$, $Cx \geq d$.

Time varying optimization Another very interesting direction for future research is tackling time-varying optimization problems in a distributed set-up (see for example Simonetto and Leus (2014)). Some of the convergence proofs provided in this thesis are based on rewriting the algorithm as a time-varying system and then on using tools to prove its stability. As so, it is possible to envision the prospect of dealing with optimization problems that varies over time. Time varying problems can arise for example in control.

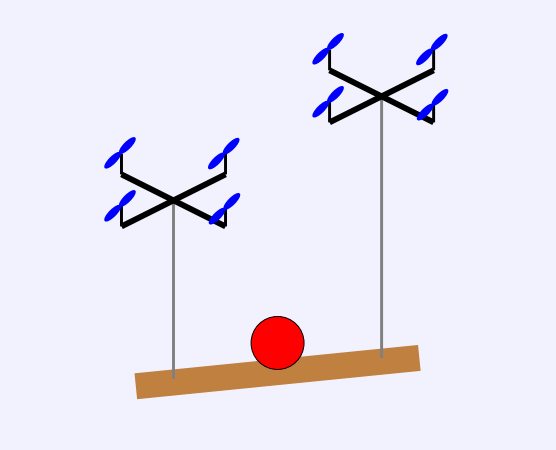


Figure 7.1: Ball and beam system supported by quadcopters.

Imagine a ball and beam problem, with the beam supported on its two extremes by two quadcopters, as in Figure 7.1.

The problem of keeping the ball on the beam, controlling the movements of the quadcopters can be formulated as an optimization problem. Nowadays, a problem like this is managed and solved in a centralized way using an MPC approach. However, it might be interesting to make the two quadcopters solve the problem in a distributed way without relying on a centralized control. Being able to manage the optimization considering also possible packet losses is of paramount importance, since the control action has to be chosen very frequently (and there might be no time to wait for the ACKs). The example reported is quite simple, but it is possible to imagine more complicated optimal control problems solvable in a distributed way and which have to cope with non-ideal communication.

A

Mathematical preliminaries, symbols and notation

This appendix provides a list of symbols, definitions and notions used along the thesis. They are divided in macro-areas in order to help the reader.

Sets

$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{Z}_{\geq 0}$	set of non-negative integer numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}_{>0}$	set of positive real numbers
$\mathbb{C}$	set of complex numbers

Uppercase calligraphic symbols, e.g. $\mathcal{A}, \mathcal{V}, \mathcal{E}$, denote sets.

$\lvert\mathcal{A}\rvert$	cardinality of set $\mathcal{A}$
$\mathcal{B} \subseteq \mathcal{A}$	set $\mathcal{B}$ is a (non necessarily proper) subset of $\mathcal{A}$
$\mathcal{A} \cup \mathcal{B}$	union of the elements of $\mathcal{A}$and $\mathcal{B}$
$[a, b]$	real interval between $a \in \mathbb{R}$ and $b \in \mathbb{R}$, $a < b$

Scalar, vectors, matrices and related operators

Lowercase italic letters, e.g. x, v, z, denote scalar values.

$\lvert a \rvert$	absolute value of $a \in \mathbb{R}$

Lowercase bold italic letters, e.g. $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}$, denote vectors of real elements, that is $\boldsymbol{x} \in \mathbb{R}^n$. Lowercase bold gothic letters, e.g. $\mathfrak{v}, \mathfrak{i}$, denote vectors of complex elements, that is $\mathfrak{v} \in \mathbb{C}^n$.
Given a collection of n scalars $x_1, \ldots, x_n$, to collect these scalars into vector $\boldsymbol{x}$ means to define the following

$$\boldsymbol{x} := \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

The $i-$th element of vector $\boldsymbol{x}$ is denoted as x_i or as $[\boldsymbol{x}]_i$.
A vector $\boldsymbol{x}$ is *strictly positive* if $x_i > 0,\ \forall i \in \{1, \ldots, n\}$.

$\mathbf{1}_n$	all-1s vector of dimension n
$\mathbf{0}$	vector of all-0s (dimension is not specified)
$\boldsymbol{x}^\top$	transpose of vector $\boldsymbol{x} \in \mathbb{R}^n$
$\mathrm{diag}(\boldsymbol{x})$	$n \times n$ diagonal matrix with diagonal elements $x_1, \ldots, x_n$
$\Re(\mathfrak{v})$	real part of $\mathfrak{v} \in \mathbb{C}^n$
$\Im(\mathfrak{v})$	imaginary part of $\mathfrak{v} \in \mathbb{C}^n$
$\lVert \boldsymbol{x} \rVert$	2-norm of vectos $\boldsymbol{x} \in \mathbb{R}^n$, $\lVert \boldsymbol{x} \rVert_2$
$\boldsymbol{x} \odot \boldsymbol{y}$	Hadamard (component-wise) product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$\boldsymbol{x}/\boldsymbol{y}$	Hadamard (component-wise) division of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$

Uppercase italic letters denote, e.g. A, B, X, denote real matrices, that is $A \in \mathbb{R}^{n \times m}$. Uppercase gothic letters, e.g. $\mathfrak{L}$, denote complex matrices, that is $\mathfrak{L} \in \mathbb{C}^{n \times m}$.
A square matrix $A \in \mathbb{R}^{n \times n}$ is *positive definite* if all its eigenvalues are strictly positive.
A matrix $A \in \mathbb{R}^n$ is *symmetric* if its equal to its transpose.
A matrix $A \in \mathbb{R}^{n \times n}$ with non-negative elements is *primitive* if there exists $1 \leq k \leq n$ such that A^k has all positive elements.
A matrix $P \in \mathbb{R}^{n \times n}$ is *stochastic* if it has non-negative elements and $P\mathbf{1}_n = \mathbf{1}_n$. P is *doubly-stochastic* if $\mathbf{1}_n^\top P = \mathbf{1}_n^\top$. A stochastic matrix P has an eigenvalue equal to 1. The eigenvalues $\lambda_1, \ldots, \lambda_n$ of a stochastic, symmetric and primitive matrix P are real and respect $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_n > -1$. The quantity $\max\{|\lambda_2|, |\lambda_n|\} < 1$ is the *essential*

spectral radius (ESR) of such a matrix. With a little abuse of notation, given a matrix which has an eigenvalue in 1 which is also the largest eigenvalue in modulus, its second largest eigenvalue in absolute value is again called ESR.

I_n	identity matrix of dimension n
$\mathbf{0}$	matrix of all-0s (dimension is not necessarily specified)
$A^\top$	transpose of matrix $A \in \mathbb{R}^{n\times m}$
$[A]_{ij}$	element in position (i,j) of matrix A
A^{-1}	inverse matrix of $A \in \mathbb{R}^{n\times n}$
$\mathrm{diag}(A)$	diagonal matrix with the same diagonal elements of matrix $A \in \mathbb{R}^{n\times n}$
$A \succ \mathbf{0}$	matrix $A \in \mathbb{R}^{n\times n}$ is positive definite
$A \succ B$	$A - B \succ \mathbf{0}$, $A, B \in \mathbb{R}^{n\times n}$

Graphs

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{V} = \{1, \dots, N\}$	set of nodes
$\mathcal{E} = \{(i,j) \,\vert\, i,j \in \mathcal{V}, i \neq j\} \cup \{(i,i) \,\vert\, i \in \mathcal{V}\}$	set of edges (contains self loops)

A graph is *directed* if $(i,j) \in \mathcal{E}$ does not imply $(j,i) \in \mathcal{E}$.
Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a *directed path* from ℓ to j consists of a sequence of vertices $(i_1, i_2, \dots, i_r)$ such that $i_1 = \ell,\ i_r = j,\ (i_j, i_{j+1}) \in \mathcal{E}$ for every $j \in \{1, \dots, r-1\}$.
A directed graph is strongly connected if for each pair $i,\ j \in \mathcal{V}$ there exists a directed path from i to j and vice-versa.

$\mathcal{N}_i^{\text{in}} = \{j \,\vert\, j \in \mathcal{V}\,, i \neq j,\ (j,i) \in \mathcal{E}\}$	in-neighbors of node i (i is not considered)
$\mathcal{N}_i^{\text{out}} = \{j \,\vert\, j \in \mathcal{V}\,, i \neq j,\ (i,j) \in \mathcal{E}\}$	out-neighbors of node i (i is not considered)

A graph is *undirected* if $(i,j) \in \mathcal{E}$ implies $(j,i) \in \mathcal{E}$. As a consequence $\mathcal{N}_i^{\text{in}} = \mathcal{N}_i^{\text{out}}$.
A *path* in $\mathcal{G}$ from ℓ to j consists of a sequence of vertices $(i_1, i_2, \dots, i_r)$ such that either $i_1 = \ell,\ i_r = j,\ (i_j, i_{j+1}) \in \mathcal{E}$ or $(i_{j+1}, i_j) \in \mathcal{E}$ for every $j \in \{1, \dots, r-1\}$.
An undirected graph $\mathcal{G}$ is *connected* if for each pair $i, j \in \mathcal{V}$ there exists a path from i to j.
A matrix $Q \in \mathbb{R}^{N\times N}$ is said to be *consistent with graph* $\mathcal{G}$ if $[Q]_{ij} > 0 \Leftrightarrow (i,j)$ belongs to $\mathcal{E}$. Given an undirected and connected graph $\mathcal{G}$, a matrix Q consistent with $\mathcal{G}$ is primitive. For such a graph, a stochastic matrix P consistent with graph $\mathcal{G}$ can be chosen symmetric. As a consequence, this P is stochastic, symmetric and primitive and its eigenvalues have

the properties described before (in particular $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_N > -1$).
For an undirected graph these additional symbols for the neighbors are employed

$\mathcal{N}_i = \{j \,\vert\, j \in \mathcal{V}, i \neq j, (j,i) \in \mathcal{E}\}$	neighbors of node i (i is not considered)
$\mathcal{N}_i^+ = \{j \,\vert\, j \in \mathcal{V}, (i,j) \in \mathcal{E}\}$	neighbors of node i, i is considered

The adjacency matrix $A_\mathcal{G}$ of a (either directed or undirected) graph $\mathcal{G}$ is an $N \times N$ matrix with $[A_\mathcal{G}]_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and 0 otherwise.
An undirected graph is *regular* if $|\mathcal{N}_i| = |\mathcal{N}_j|$ for all $i, j \in \mathcal{V}$.
A (directed or undirected) graph is *complete* if the edge set $\mathcal{E}$ contains all the possible pair $(i,j) \in \mathcal{V} \times \mathcal{V}$.
A (directed or undirected) graph is *bipartite* if it is possible to partition the node set $\mathcal{V}$ in two parts, $\mathcal{V}_1$ and $\mathcal{V}_2$ such that all the edges connect nodes of $\mathcal{V}_1$ to nodes of $\mathcal{V}_2$ and vice-versa (the self loops however are still considered), and there are no edges connecting nodes of $\mathcal{V}_1$ (of $\mathcal{V}_2$) to nodes of $\mathcal{V}_1$ (of $\mathcal{V}_2$ resp.).
In a *random geometric* graph with N nodes, the nodes are randomly arranged in a squared environment of edge equal to 1. The distance threshold $r > 0$ determines the edge set, according to the following rule: agent i and j are connected (that is (i,j) and (j,i) belong to $\mathcal{E}$) if and only if their distance is smaller or equal to r.

Functions

$\mathcal{C}^s$	set of s times continuously differentiable functions
$f'(x)$	first derivative of function $f : \mathbb{R} \mapsto \mathbb{R}$ evaluated at x
$f''(x)$	second derivative of function $f : \mathbb{R} \mapsto \mathbb{R}$ evaluated at x
$\nabla f(\boldsymbol{x})$	gradient of function $f : \mathbb{R}^n \mapsto \mathbb{R}$ evaluated at $\boldsymbol{x}$
$\nabla^2 f(\boldsymbol{x})$	Hessian of function $f : \mathbb{R}^n \mapsto \mathbb{R}$ evaluated at $\boldsymbol{x}$

A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *convex* if for all $\boldsymbol{x}, \boldsymbol{y}$ in $\mathbb{R}^n$ and $0 \leq \theta \leq 1$ it holds

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}).$$

Convex functions have very interesting properties, that will be used throughout the dissertation. The interested reader is referred to Boyd and Vandenberghe (2004) for a better understanding of these properties.

Random variables

$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	normal random vector of mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and variance $\Sigma \in \mathbb{R}^{n \times n}$
$\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$	$\boldsymbol{v}$ is a realization of $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$

Time

Concerning the concept of time, it is assumed that the local variables at each node are updated at discrete time instants (e.g., based on local and possibly unsynchronized clocks, or based on events like receiving a packet). From a global perspective, all time instants when at least one variable in one node is updated are collected and ordered in the sequence $\{t_k\}_{k=1}^{\infty}$. With a little abuse of notation, a variable x at time t_k (which is one of the times when some variables are being modified) is denoted as $x(k)$ instead of $x(t_k)$ and the evolution of the nodes' variables is studied as a discrete-time system. Also, sometime the time instant t_k is denoted as k, even though this is not formally correct.

B

Appendix for Chapter 3

B.1 Proof of Theorem 3.3.1

Point (i) can be easily verified by analysing the steps of the algorithm. For **point (ii) and (iii)** only the part concerning J_2 is proven (the part of point (ii) related to J_∞ follows from Theorem 3.4.1). Observe that after τ iterations of the BC algorithm, $r_i(k) = \ell_{i+1}(k)$ for all $i = 1, \dots, N$. For $k \geq \tau$, it is then possible to introduce the auxiliary variables $x_i(k) = r_i(k) = \ell_{i+1}(k)$, $i = 1, \dots, N-1$, and let $\boldsymbol{x}(k)$ be the vector collecting all $x_i(k)$. Looking at the iterations of the algorithm as the evolution of a dynamical system, vector $\boldsymbol{x}(k)$ represents the state of this system. The goal is to apply the following theorem:

Theorem B.1.1. *[Theorem 4.3 in Bullo et al. (2012)]*

Let (X, d) be a metric space. Given a collection of maps $T_1, \dots, T_N : X \mapsto X$, define the set-valued map $T : X \rightrightarrows X$ by $T(x) = \{T_1(x), \dots, T_N(x)\}$ and let $\{x_k\}_{k \in \mathbb{Z}_{\geq 0}}$ be an evolution of T. Assume that

- *There exists a compact set $W \subseteq X$ that is strongly positive invariant for T;*
- *There exists a function $U : W \mapsto \mathbb{R}$ such that $U(w') < U(w)$ for all $w \in W$ and $w' \in T(w) \setminus \{w\}$;*

- *The maps T_i for $i \in \{1, \dots, N\}$ and U are continuous on W;*
- *For all $i \in \{1, \dots, N\}$, there exists an increasing sequence of times $\{k_t \mid t \in \mathbb{Z}_{\geq 0}\}$ such that $x_{k_t+1} = T_i(x_{k_t})$ and $(k_t + 1 - k_t)$ is bounded.*

If $x_0 \in W$, there exists $c \in \mathbb{R}$ such that the evolution $\{x_k\}_{k \in \mathbb{Z}_{\geq 0}}$ approaches the set

$$(F_1 \cap \cdots \cap F_N) \cap U^{-1}(c),$$

where $F_i = \{w \in W \mid T_i(w) = w\}$ is the set of fixed points of T_i in W, $i \in \{1, \dots, N\}$.

It is therefore necessary to verify that all the hypotheses of this theorem are satisfied.

First of all, observe that, according to the physical constraints, $\underline{d}_{i+1} \leq x_i(k) \leq \overline{d}_i$ and so $\boldsymbol{x}(k)$ can take values only in $W = \prod_{i=1}^{N-1} \left[\underline{d}_{i+1}, \overline{d}_i\right]$. Since W is given by the Cartesian product of $N-1$ closed intervals, it follows that W is compact. Next, for $i \in \{1, \dots, N\}$, let $T_i : W \to W$ be the map describing the updating iteration of CB algorithm in case camera i is the camera performing the forward communication round. Observe that, for $i \in \{1, \dots, N\}$, the map T_i is continuous with respect to the standard Euclidean metric. Now, for $\boldsymbol{x} = [x_1, \dots, x_{N-1}] \in W$, consider the function $U : W \to \mathbb{R}$ such that

$$U(\boldsymbol{x}(k)) = \frac{1}{2} \sum_{i=1}^{N} \frac{L_i(k)^2}{\bar{v}_i}, \tag{B.1}$$

where $L_i(k) = r_i(k) - \ell_i(k) = x_i(k) - x_{i-1}(k)$. It is necessary to show that U is a Lyapunov function for the update of the algorithm, i.e, that $U(\boldsymbol{x}(k+1)) < U(\boldsymbol{x}(k))$ whenever $\boldsymbol{x}(k+1) \neq \boldsymbol{x}(k)$. To prove this it is useful to introduce the following

FACT. *Let L, α, β be three positive real numbers. Then the minimizer of the function $g(z) = \frac{z^2}{\alpha} + \frac{(L-z)^2}{\beta}$ within the interval $[0, L]$ is given by $x = \frac{\alpha L}{\alpha+\beta}$.*

Suppose that at time k the $i-$th camera is activated, $i \neq 1$ and $i \neq N$, and consider the following sum of terms:

$$\begin{aligned}
\Gamma =& \frac{1}{2}\frac{L_{i-1}(k)^2}{v_{i-1}} + \frac{1}{2}\frac{L_i(k)^2}{v_i} + \frac{1}{2}\frac{L_{i+1}(k)^2}{v_{i+1}} = \\
=& \frac{1}{v_{i-1}}\left(\frac{L_{i-1}(k)}{2}\right)^2 + \underbrace{\frac{1}{v_{i-1}}\left(\frac{L_{i-1}(k)}{2}\right)^2 + \frac{1}{v_i}\left(\frac{L_i(k)}{2}\right)^2}_{\gamma_1} + \\
&+ \underbrace{\frac{1}{v_i}\left(\frac{L_i(k)}{2}\right)^2 + \frac{1}{v_{i+1}}\left(\frac{L_{i+1}(k)}{2}\right)^2}_{\gamma_2} + \frac{1}{v_{i+1}}\left(\frac{L_{i+1}(k)}{2}\right)^2.
\end{aligned}$$

Recalling the quantities in Formulas (3.6) and the value of $c_\ell^*(k)$, considering $\tilde{L}_{i-1}(k) = \frac{L_{i-1}(k)}{2} + \frac{L_i(k)}{2} = m_i(k) - m_{i-1}(k)$, the point $c_\ell^*(k) - m_{i-1}(k)$ is the minimizer of function

$g(z)$ of parameters $L = \tilde{L}_{i-1}(k)$, $\alpha = v_{i-1}$ and $\beta = v_i$, as can be verified by calculation. Introduce now $L'_{i-1}(k) = r_{i-1}(k+1) - m_{i-1}(k)$ and $L'_i(k) = m_i(k) - r_{i-1}(k+1) = \tilde{L}_{i-1}(k) - L'_{i-1}(k)$.
Since that $r_{i-1}(k) \leq r_{i-1}(k+1) \leq c^*_\ell$ or $r_{i-1}(k) \geq r_{i-1}(k+1) \geq c^*_\ell$, the update implies that

$$\gamma_1 = g\left(\frac{L_{i-1}(k)}{2}\right) \geq g(L'_{i-1}(k)).$$

A similar reasoning holds considering the update of camera $i+1$. According to the latter, defining $L''_{i+1}(k) = m_{i+1}(k) - \ell_{i+1}(k+1)$ and $L''_i(k) = \ell_{i+1}(k+1) - m_i(k)$, it holds

$$\gamma_2 \geq \frac{1}{v_i}(L''_i(k))^2 + \frac{1}{v_{i+1}}(L''_{i+1}(k))^2.$$

As a consequence,

$$\Gamma \geq \underbrace{\frac{1}{v_{i-1}}\left(\frac{L_{i-1}(k)}{2}\right)^2 + \frac{1}{v_{i-1}}(L'_{i-1}(k))^2}_{\delta_1} +$$
$$+\underbrace{\frac{1}{v_i}(L'_i(k))^2 + \frac{1}{v_i}(L''_i(k))^2}_{\delta_2} + \underbrace{\frac{1}{v_{i+1}}(L''_{i+1}(k))^2 + \frac{1}{v_{i+1}}\left(\frac{L_{i+1}(k)}{2}\right)^2}_{\delta_1}.$$

Now define $L_{i-1}(k+1) = \frac{L_{i-1}(k)}{2} + L'_{i-1}(k)$, $L_{i+1}(k+1) = \frac{L_{i+1}(k)}{2} + L''_{i+1}(k)$ and $L_i(k+1) = L'_i(k) + L''_i(k)$. Analysing δ_1, it holds that

$$\delta_1 = \frac{1}{v_{i-1}} g\left(\frac{L_{i-1}(k)}{2}\right) \geq \frac{1}{v_{i-1}} g\left(\frac{L_{i-1}(k+1)}{2}\right)$$

where in this case the parameters of function g are $L = L_{i-1}(k+1)$, $\alpha = \beta = 1$. A similar reasoning holds for δ_2 and δ_3. All the previous considerations lead to the following

$$U(x(k)) \geq \sum_{\substack{j=1 \\ j \neq i, i-1, i+1}}^{N} \frac{L_j(k)^2}{2} + 2\frac{1}{v_{i-1}}\left(\frac{L_{i-1}(k+1)}{2}\right)^2 +$$
$$+ 2\frac{1}{v_i}\left(\frac{L_i(k+1)}{2}\right)^2 + 2\frac{1}{v_{i+1}}\left(\frac{L_{i+1}(k+1)}{2}\right)^2 = U(x(k+1).$$

When the camera that is activated at time k is the 1-st or the $N-$th a similar reasoning shows that $U(x(k)) \geq U(x(k+1))$. Therefore, $U(x(k)) \geq U(x(k+1))$, and the inequality is strict as long as at least one of the following holds, $l_{i+1}(k+1) \neq l_{i+1}(k)$, or $r_{i-1}(k+1) \neq r_{i-1}(k)$.

It is now possible to apply Theorem 4.3 of Bullo et al. (2012) and to conclude that $\boldsymbol{x}(k)$ converges to the set $F_1 \cap \cdots \cap F_N \cap U^{-1}(c)$ for some $c \in \mathbb{R}$, where $F_i = \{\boldsymbol{x} \in W \mid T_i(\boldsymbol{x}) = \boldsymbol{x}\}$ is the set of fixed points of T_i. Finally, proving that $F_1 \cap \cdots \cap F_N$ is a singleton concludes the proof (since if $F_1 \cap \cdots \cap F_N = \tilde{\boldsymbol{x}}$, then $c = U(\tilde{\boldsymbol{x}})$).

To do so, recall that J_2 has a unique minimizer $\boldsymbol{\xi}^*$, and so the corresponding $\boldsymbol{x}^*$ is a fixed point of all the maps T_i (note that it is possible to find a bijective correspondence between an $\boldsymbol{x}$ and a $\boldsymbol{\xi}$). This follows from the fact that, since $U(\boldsymbol{x}(k)) = \frac{1}{2}J_2(\boldsymbol{\xi}(k))$, what was just showed proves that if there exists T_i such that $T_i(\boldsymbol{\xi}^*) \neq \boldsymbol{\xi}^*$ then $J_2(\boldsymbol{\xi}^*) > J_2(T_i(\boldsymbol{\xi}^*))$, and this is a contradiction. Assume now that there exists $\boldsymbol{\xi}' \neq \boldsymbol{\xi}^*$ such that $\boldsymbol{\xi}' \in (F_1 \cap \cdots \cap F_N)$. Since $\boldsymbol{\xi}^*$ is the unique minimizer of J_2, it holds $J_2(\boldsymbol{\xi}^*) < J_2(\boldsymbol{\xi}')$, but a contradiction arises because $\boldsymbol{\xi}' \in (F_1 \cap \cdots \cap F_N)$ implies that there is no possibility to improve the cost function $U(\boldsymbol{x}')$. Therefore $F_1 \cap \cdots \cap F_N$ is a singleton that coincides with the minimizer of problem $\mathcal{P}_3$ and so $\boldsymbol{\xi}(k)$ converges to $\boldsymbol{\xi}^*$. Finally, since $J_2(k)$ converges to J_2^*, necessarily also $J_\infty(k)$ converges to J_∞^*.

B.2 Proof of Theorem 3.4.1

The following theorem is a refinement of Theorem 4.3 in Bullo et al. (2012), valid for a specific class of dynamical switching systems.

Theorem B.2.1. *Let $W \subset \mathbb{R}^n$ be a compact set. Let M be a (finite) positive integer and let $\{T_i : W \to W,\ i = 1, \ldots, M\}$ be a set of M functions. Assume that*

- *There exists a function $J : W \to \mathbb{R}$ such that*

$$J\left(T_i(x)\right) \leq J(x), \qquad \forall\, x \in W, \tag{B.2}$$

$$J(T_i(x)) < J(x), \qquad \forall\, x \notin \mathcal{W}^*, \tag{B.3}$$

  where $\mathcal{W}^*$ is the minimum value attained by J over W;

- *The maps T_i for $i\,\{1, \ldots, M\}$ and J are continuous on W;*

Consider the trajectory generated by

$$\boldsymbol{x}(k+1) = T_{\sigma(k)}(\boldsymbol{x}(k)), \qquad \boldsymbol{x}(0) \in W,$$

where $\sigma : \mathbb{Z}_{\geq 0} \to \{1, \ldots, m\}$ is a process determining which map within the set $\{T_1, \ldots, T_M\}$ is selected at iteration k. Then, if J^* is the minimum value of J over the set W,

$$\lim_{t \to \infty} J\left(x(k)\right) = J^*$$

and $\boldsymbol{x}(k)$ *converges to the set* $\mathcal{W}^*$.

Proof. The proof follows using the same continuity arguments adopted in the proof of Theorem 4.3 in Bullo et al. (2012). ■

Proof of Theorem 3.4.1. **Point (i)** is an immediate consequence of the steps of the algorithm. Concerning **Point (ii)**, denote $T_{lag}(A_i(k))$ as $T^i_{lag}(k)$, and denote $T_{\max}(k) := \max_i \left\{T^i_{lag}(k)\right\}$. It is possible to state these two preliminary facts.
FACT I. *If camera* i *successfully transmits to camera* $i+1$ *at time* k*, and if* $T^i_{lag}(k) > T^{i+1}_{lag}(k)$*, then* $T^{i+1}_{lag}(k+1) < T^i_{lag}(k)$*. As a consequence* $T^{i+1}_{lag}(k') < T^{\max}_{lag}(k)$, $\forall k' > k$.

To confirm the validity of the above fact observe that, due to the algorithm step, it holds

$$T^{i+1}_{lag}(k+1) \leq 2\left(\frac{3}{8}T^{i+1}_{lag}(k) + \frac{T^i_{lag}(k)}{8}\right) < T^i_{lag}(k).$$

Since $T^i_{lag}(k) \leq T_{\max}(k)$, the last sentence follows by induction.
FACT II. *If camera* $i+1$ *successfully transmits to camera* i *at time* k*, and if* $T^i_{lag}(k) > T^{i+1}_{lag}(k)$ *and* $\ell_{i+1}(k) < r_i(k)$*, then* $T^i_{lag}(k+1) < T^i_{lag}(k)$.

Since $T^i_{lag}(k) > T^{i+1}_{lag}(k)$, the algorithm tries to diminish $T^i_{lag}(k)$. The fact that $\ell_{i+1}(k) < r_i(k)$ allows to argue that $r_i(k+1) < r_i(k)$. The statement easily follows.

Now, observe that from Fact I it follows that $J_\infty(k)$ is non increasing. To prove that $J_\infty(k+\tau_{\max}) < J_\infty(k)$ if $\boldsymbol{\xi}(k) \notin \Xi^*_{\mathcal{P}_1}$,first suppose that only camera i is such that $T^i_{lag}(k) = T_{\max}(k)$. Since $\boldsymbol{\xi}(k) \notin \Xi^*_{\mathcal{P}_1}$, it holds that $T^i_{lag}(k) = T_{\max}(k) > T^*_{\mathcal{P}_1}$. As a consequence it is not possible to have that both, $r_{i-1}(k) = \bar{d}_{i-1}$ and $\ell_{i+1}(k) = \underline{d}_{i+1}$. Suppose also that $\ell_{i+1}(k) = r_i(k)$ and $\ell_{i+1}(k) > \underline{d}_{i+1}$ (all the other starting situations lead to the same conclusion). Due to the assumptions, defining $\tilde{\tau} = h\tau$, there exists a $\tilde{k}$, $k \leq \tilde{k} \leq k + \tilde{\tau}$ such that camera i successfully communicates with camera $i+1$. As a consequence, $\ell_{i+1}(\tilde{k}+1) < r_i(k)$ due to Fact I. If the backward communication works, $T^i_{lag}(\tilde{k}+1) < T^i_{lag}(k) = T_{\max}(k)$ and the result follows. Otherwise in $[\tilde{k}+1, \tilde{k}+\tilde{\tau}]$ there is a working forward communication between cameras $i+1$ and i, for which (due to Fact I) the hypothesis of Fact II hold. As a consequence, for sure $T^i_{lag}(t+2\tilde{\tau}+1) < T^i_{lag}(k) = T_{\max}(k)$.

If there is more than one camera i such that $T^i_{lag}(k) = T_{\max}(k)$, it is possible to show using the previous reasoning that $J_\infty(k + 2\tilde{\tau}(N-1)+1) < J_\infty(k)$. This is the time interval required for the two worst possible cases: one of these is when at time k cameras $1, \ldots, N-1$ have time lag $T_{\max}(k)$, are such that $r_i(k) = \ell_{i+1}(k), i = 1 \ldots, N-1$, and only the last camera has a time lag smaller than $T_{\max}(k)$ (the other case is the one with

cameras $2, \dots, N$ that have time lag $T_{\max}(k)$). Defining $\tau_{\max} := 2\tilde{\tau}(N-1)+1$ the result is proven.

Finally, concerning **point (iii)**: consider vector $\boldsymbol{\xi}(k) \in \mathbb{R}^{2N}$ associated to $\{A_i(k)\}_{i=1}^{N}$, and its reduced version $\boldsymbol{\xi}'(k) \in \mathbb{R}^{2N-2}$ corresponding to $\boldsymbol{\xi}(k)$ without its first and last elements (that are always 0 and L respectively). Consider the sequence $\{\boldsymbol{x}_t\}_{t=1}^{\infty}$ that represents the evolution of the patrolling areas given by the algorithm every $\tau_{\max}$ instants, i.e. $\boldsymbol{x}_1 = \boldsymbol{\xi}'(1)$ and $\boldsymbol{x}_t = \boldsymbol{\xi}'(1+(t-1)\tau_{\max}),\ t > 1$. Due to the physical bounds of the cameras, $\boldsymbol{x}_t$ belongs to the compact set W obtained as the Cartesian product of intervals (as done for the previous proof).

Now, define maps $T_1, \dots, T_M$, with M a finite integer, in the following way: there exists a map $T_j : W \to W$ for every possible camera activation sequence of length $\tau_{\max} - 1$ and the related communications that work for each activation, respecting both Assumptions 2.3.2 and 2.3.1 (note that these maps are different with respect to those introduced in the previous proof). In this way, it is always possible to find a $j \in \{1, \dots, M\}$ such that $\boldsymbol{x}_{t+1} = T_j(\boldsymbol{x}_t)$. Since each possible step of the algorithm is a continuous function, also every T_i is a continuous function.

Note now that J_∞ is a continuous function such that $J_\infty(T_j(\boldsymbol{x}_t)) \leq J_\infty(\boldsymbol{x}\_t),\ \forall \boldsymbol{x}_t \in W$, $j \in \{1, \dots, M\}$, and $J_\infty(T_j(\boldsymbol{x}_t)) < J_\infty(\boldsymbol{x}_t),\ \forall \boldsymbol{x}_t \notin \Xi^*_{\mathcal{P}_1}, j \in \{1, \dots, M\}$ due to point (ii). Using Theorem B.2.1, $J_\infty(\boldsymbol{x}_t)$ converges to J^*_∞. Since at each iteration of the *r-CB* algorithm the cost function is smaller or equal to the previous step, $J_\infty(k)$ converges to J^*_∞. ■

C

Appendix for Chapter 4

C.1 Proof of Theorem 4.5.3

The proof of Theorem 4.5.3 relies on the time scale separation of the dynamic of the $\boldsymbol{x}_i$s and of the auxiliary variables $\widehat{\boldsymbol{x}}_j^{(i)}$s, $\widehat{\boldsymbol{\rho}}_j^{(i)}$s and $\widehat{\xi}_j^{(i)}$s, and fully exploits the following Lemma

Lemma C.1.1 (Time scale separation principle for discrete time dynamical systems)**.** *Consider the dynamical system*

$$\begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{y}(k+1) \end{bmatrix} = \begin{bmatrix} I & -\epsilon B \\ C(k) & F(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{bmatrix}. \tag{C.1}$$

Let the following assumptions hold

1. *There exists a matrix G such that $\boldsymbol{y} = G\boldsymbol{x}$ satisfies the expression $\boldsymbol{y} = C(k)\boldsymbol{x} + F(k)\boldsymbol{y},\ \forall k, \forall \boldsymbol{x}$*

2. *the system*

$$\boldsymbol{z}(k+1) = F(k)\boldsymbol{z}(k) \tag{C.2}$$

is exponentially stable;

3. *the system*

$$\dot{\boldsymbol{x}}(k) = -BG\boldsymbol{x}(k) \tag{C.3}$$

is exponentially stable.

4. *The matrices $C(k)$ and $F(k)$ are bounded, i.e. there exists $m > 0$ such that $\|C(k)\| < m, \|F(k)\| < m, \forall k \geq 0$.*

Then, there exists $\bar{\epsilon}$, with $0 < \epsilon < \bar{\epsilon}$ such that the origin is an exponentially stable equilibrium for the system (C.1). □

Proof of Lemma C.1.1. First consider the following change of variables:

$$\boldsymbol{z}(k) = \boldsymbol{y}(k) - G\boldsymbol{x}(k)$$

The dynamics of the system in the variables $\boldsymbol{x}, \boldsymbol{z}$ can be written after some straightforward manipulations as follows:

$$\begin{bmatrix}\boldsymbol{x}(k+1)\\ \boldsymbol{z}(k+1)\end{bmatrix} = \left(\underbrace{\begin{bmatrix}I - \epsilon BG & 0\\ 0 & F(k)\end{bmatrix}}_{\Sigma(k)} + \epsilon \underbrace{\begin{bmatrix}0 & -BG\\ GBG & GB\end{bmatrix}}_{\Gamma}\right)\underbrace{\begin{bmatrix}\boldsymbol{x}(k)\\ \boldsymbol{z}(k)\end{bmatrix}}_{\mu(k)} \tag{C.4}$$

where Assumption 1 was used. From Assumption 2 and 3, using converse Lyapunov theorems Khalil (2001), it follows that there exist positive definite matrices $P_{\boldsymbol{x}} > 0$ and $P_{\boldsymbol{z}}(k) > 0$ such that

$$-P_{\boldsymbol{x}}BG - G^T B^T P_{\boldsymbol{x}} \leq -aI, \quad F(k)^T P_{\boldsymbol{z}}(k+1)F(k) - P_{\boldsymbol{z}}(k) \leq -aI, \forall k$$

where a is a positive scalar and $P_{\boldsymbol{z}}(k)$ is bounded, i.e. $\|P_{\boldsymbol{z}}(k)\| \leq m$. The following positive definite Lyapunov function is useful to prove exponential stability of the whole system:

$$U(\boldsymbol{x}, \boldsymbol{z}, k) = \boldsymbol{x}^T P_{\boldsymbol{x}} \boldsymbol{x} + \boldsymbol{z}^T P_{\boldsymbol{z}}(k)\boldsymbol{z} = \begin{bmatrix}\boldsymbol{x}^T & \boldsymbol{z}^T\end{bmatrix} \underbrace{\begin{bmatrix}P_{\boldsymbol{x}} & 0\\ 0 & P_{\boldsymbol{z}}(k)\end{bmatrix}}_{P(k)} \begin{bmatrix}x\\ z\end{bmatrix}$$

Defining the time difference of the Lyapunov function as $\Delta U(\boldsymbol{x}, \boldsymbol{z}, k) = U(\boldsymbol{x}(k+1), \boldsymbol{z}(k+$

$1), t+1) - U(\boldsymbol{x}(k), \boldsymbol{z}(k), k)$ it holds:

$$\begin{aligned}\Delta U(\boldsymbol{x}, \boldsymbol{z}, k) \;=\; & \boldsymbol{x}^T \left(-\epsilon(P_{\boldsymbol{x}}BG + G^T B^T P_{\boldsymbol{x}}) + \epsilon^2 G^T B^T P_{\boldsymbol{x}} BG\right) \boldsymbol{x} + \\ & + \boldsymbol{z}^T \Big(F(k)^T P_{\boldsymbol{z}}(k+1)F(k) - P_{\boldsymbol{z}}(k)\Big)\, \boldsymbol{z} + 2\epsilon\mu^T \Sigma^T(k) P(k+1)\Gamma\mu + \epsilon^2 \mu^T \Gamma^T P(k+1)\Gamma\mu \\ \leq\; & -\epsilon a \|\boldsymbol{x}\|^2 - a\|\boldsymbol{z}\|^2 + \epsilon^2 \underbrace{\|P_{\boldsymbol{x}}^{\frac{1}{2}} BG\|^2}_{b} \|\boldsymbol{x}\|^2 + 2\epsilon\mu^T \Sigma^T(k) P(k+1)\Gamma\mu + \epsilon^2 \|P^{\frac{1}{2}}(k+1)\Gamma\|^2 \|\mu\|^2\end{aligned}$$

Note that the top left block of Γ is zero and that $\Sigma(k)$ and $P(k)$ are diagonal and bounded for all times. From this it follows that

$$\Sigma^T(k)P(k+1)\Gamma = \begin{bmatrix} 0 & \star \\ \star & \star \end{bmatrix} \Longrightarrow 2\mu^T \Sigma^T(k) P(k+1)\Gamma\mu \leq c(2\|\boldsymbol{x}\|\|\boldsymbol{z}\| + \|\boldsymbol{z}\|^2)$$

for some positive scalar c. Boundedness of $P(k)$ also implies that

$$\|P^{\frac{1}{2}}(k+1)\Gamma\|^2 \|\mu\|^2 \leq d(\|\boldsymbol{x}\|^2 + \|\boldsymbol{z}\|^2)$$

for some positive scalar d. Putting all together we get

$$\Delta U(\boldsymbol{x}, \boldsymbol{z}, k) \leq \begin{bmatrix} \|\boldsymbol{x}\| & \|\boldsymbol{z}\| \end{bmatrix} \begin{bmatrix} -\epsilon a + b\epsilon^2 & \epsilon c \\ \epsilon c & -a + \epsilon c + \epsilon^2 d \end{bmatrix} \begin{bmatrix} \|\boldsymbol{x}\| \\ \|\boldsymbol{z}\| \end{bmatrix}$$

It follows immediately that there exists a critical $\bar{\epsilon}$ such that for $0 < \epsilon < \bar{\epsilon}$ the matrix in the above equation is strictly negative definite and therefore the system is exponentially stable. ■

It is now possible to state the formal proof of Theorem 4.5.3.

Proof of Theorem 4.5.3. The proof relies on Lemma C.1.1. In order to improve readability, the proof is broken into few steps. The first step is to write the evolution of the RBJ algorithm as the evolution of a dynamical system. The second step is to find its equilibrium point and to linearize it around this point. The third step is to show that the linerized dynamical system satisfies the three assumptions listed in Lemma C.1.1.

RBJ as a dynamical system:
First of all, note that thanks to Assumption 4.2.2 the second order derivatives and in particular all the variables $\xi_j^{(i)}, \widehat{\xi}_j^{(i)}$ are always well defined and invertible. Now, let the vectors $\widehat{\boldsymbol{e}}_j^{(i)}$ be the *vectorization* of $\widehat{\xi}_j^{(i)}$, $\widehat{\boldsymbol{e}}_j^{(i)} = \mathrm{vec}(\widehat{\xi}_j^{(i)})$, and the *un-vectorization operator* vec^{-1} as the inverse of the vectorization operator, i.e. $\mathrm{vec}^{-1}(\widehat{\boldsymbol{e}}_j^{(i)}) = \widehat{\xi}_j^{(i)}$. Let $\widehat{\boldsymbol{x}}_i$, $\widehat{\boldsymbol{\rho}}_i$ and

$\widehat{\boldsymbol{e}}_i$ be the vectors in which all the $\widehat{\boldsymbol{x}}_j^{(i)}$'s, the $\widehat{\boldsymbol{\rho}}_j^{(i)}$'s, and the $\widehat{\boldsymbol{e}}_j^{(i)}$'s are stacked, respectively, i.e. $\widehat{\boldsymbol{x}}_i = (\widehat{\boldsymbol{x}}_{j_1}^{(i)} \cdots \widehat{\boldsymbol{x}}_{j_{\mathcal{N}_i}}^{(i)})$ and similarly for $\widehat{\boldsymbol{\rho}}_i$ and $\widehat{\boldsymbol{e}}_i$. Let $\boldsymbol{x}$, $\widehat{\boldsymbol{x}}$, $\widehat{\boldsymbol{\rho}}$, $\widehat{\boldsymbol{e}}$ be the vectors collecting all the $\boldsymbol{x}_i$, $\widehat{\boldsymbol{x}}_i$'s, $\widehat{\boldsymbol{\rho}}_i$'s and $\widehat{\boldsymbol{e}}_i$'s, respectively, i.e. $\boldsymbol{x} = (\boldsymbol{x}_1 \cdots \boldsymbol{x}_N)$ and similarly for $\widehat{\boldsymbol{x}}$, $\widehat{\boldsymbol{\rho}}$ and $\widehat{\boldsymbol{e}}$.

For every agent i and neighbours $j \in \mathcal{N}_i$, the dynamic of the local variables are given by the following equations:

$$\boldsymbol{x}_i(k+1) = f_1^i(\boldsymbol{x}(k), \widehat{\boldsymbol{\rho}}(k), \widehat{\boldsymbol{e}}(k)) \tag{C.5a}$$

$$\widehat{\boldsymbol{x}}_j^{(i)}(k+1) = f_2^{ij}(\boldsymbol{x}(k), \widehat{\boldsymbol{x}}(k), k) \tag{C.5b}$$

$$\widehat{\boldsymbol{\rho}}_j^{(i)}(k+1) = f_3^{ij}(\boldsymbol{x}(k), \widehat{\boldsymbol{x}}(k), \widehat{\boldsymbol{\rho}}(k), k) \tag{C.5c}$$

$$\widehat{\boldsymbol{e}}_j^{(i)}(k+1) = f_4^{ij}(\boldsymbol{x}(k), \widehat{\boldsymbol{x}}(k), \widehat{\boldsymbol{e}}(k), k) \tag{C.5d}$$

where

$$f_1^i(\boldsymbol{x}, \widehat{\boldsymbol{\rho}}, \widehat{\boldsymbol{e}}) = \boldsymbol{x}_i - \epsilon \underbrace{\Big(\sum_{j \in \mathcal{N}_i^+} \operatorname{vec}^{-1}(\widehat{\boldsymbol{e}}_j^{(i)}) \Big)}_{f_{\boldsymbol{e}}^i(\widehat{\boldsymbol{e}})}^{-1} \underbrace{\Big(\sum_{j \in \mathcal{N}_i^+} \widehat{\boldsymbol{\rho}}_j^{(i)} \Big)}_{f_{\boldsymbol{\rho}}^i(\widehat{\boldsymbol{\rho}})} \tag{C.6a}$$

$$f_2^{ij}(\boldsymbol{x}, \widehat{\boldsymbol{x}}, k) = \begin{cases} \widehat{\boldsymbol{x}}_j^{(i)} & \text{if } \gamma_j^{(i)}(k) = 0 \\ \boldsymbol{x}_j & \text{if } \gamma_j^{(i)}(k) = 1 \end{cases} \tag{C.6b}$$

$$f_3^{ij}(\boldsymbol{x}, \widehat{\boldsymbol{x}}, \widehat{\boldsymbol{\rho}}, k) = \begin{cases} \widehat{\boldsymbol{\rho}}_j^{(i)} & \text{if } \gamma_j^{(i)}(k) = 0 \\ \nabla_i J_j(\boldsymbol{x}_j, \{\widehat{\boldsymbol{x}}_\ell^{(j)}\}_{\ell \in \mathbb{N}_j}) & \text{if } \gamma_j^{(i)}(k) = 1 \end{cases} \tag{C.6c}$$

$$f_4^{ij}(\boldsymbol{x}, \widehat{\boldsymbol{x}}, \widehat{\boldsymbol{e}}, k) = \begin{cases} \widehat{\boldsymbol{e}}_j^{(i)} & \text{if } \gamma_j^{(i)}(k) = 0 \\ \operatorname{vec}\left(\nabla_{ii}^2 J_j(\boldsymbol{x}_j, \{\widehat{\boldsymbol{x}}_\ell^{(j)}\}_{\ell \in \mathbb{N}_j}) \right) & \text{if } \gamma_j^{(i)}(k) = 1 \end{cases} . \tag{C.6d}$$

Note that the variables $\boldsymbol{\rho}_j^{(i)}$ and $\xi_j^{(i)}$ do not appear in the dynamics since they are deterministic functions of the variables $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$, and therefore can be omitted.

Equilibrium point and linearization:

Let $\boldsymbol{x}^*$ be the minimizer of the optimization problem and define

$$
\begin{aligned}
H_{h\ell} &= \nabla^2_{h\ell} J(\boldsymbol{x}^*) = \sum_{j=1}^{N} \underbrace{\nabla^2_{h\ell} J_j(\boldsymbol{x}^*_j, \{\boldsymbol{x}^*_s\}_{s\in\mathbb{N}_j})}_{H^j_{h\ell}} = \sum_{j=1}^{N} H^j_{h\ell} \\
\widehat{\boldsymbol{x}}^{(i)*}_j &= \boldsymbol{x}^*_j \\
\widehat{\boldsymbol{\rho}}^{(i)*}_j &= \nabla_i J_j(\boldsymbol{x}^*_j, \{\boldsymbol{x}^*_\ell\}_{\ell\in\mathbb{N}_j}) \\
\widehat{\boldsymbol{e}}^{(i)*}_j &= \operatorname{vec}\left(\nabla^2_{ii} J_j(\boldsymbol{x}^*_j, \{\boldsymbol{x}^*_\ell\}_{\ell\in\mathbb{N}_j})\right) = \operatorname{vec}(H^j_{ii})
\end{aligned}
$$

Notice that $\sum_{j=1}^{N} \widehat{\boldsymbol{\rho}}^{(i)*}_j = \nabla_i J(\boldsymbol{x}^*) = 0$, since the gradient computed at the minimizer is zero. It is now simple to verify by direct inspection that $(\boldsymbol{x}^*, \widehat{\boldsymbol{x}}^*, \widehat{\boldsymbol{\rho}}^*, \widehat{\boldsymbol{e}}^*)$ is an equilibrium point for the dynamical system described by (C.5). Next, the behavior of system (C.5) in the neighborhood of the equilibrium point $(\boldsymbol{x}^*, \widehat{\boldsymbol{x}}^*, \widehat{\boldsymbol{\rho}}^*, \widehat{\boldsymbol{e}}^*)$ is analyzed. Consider the change of variables

$$
\begin{aligned}
\boldsymbol{\psi} &= \boldsymbol{x} - \boldsymbol{x}^* \\
\widehat{\boldsymbol{\psi}} &= \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}}^* \\
\widehat{\boldsymbol{\eta}} &= \widehat{\boldsymbol{\rho}} - \widehat{\boldsymbol{\rho}}^* \\
\widehat{\boldsymbol{\zeta}} &= \widehat{\boldsymbol{e}} - \widehat{\boldsymbol{e}}^*
\end{aligned}
\tag{C.7}
$$

Linearizing equations (C.5) around $(\boldsymbol{x}^*, \widehat{\boldsymbol{x}}^*, \widehat{\boldsymbol{\rho}}^*, \widehat{\boldsymbol{e}}^*)$, it holds

$$
\boldsymbol{\psi}_i(k+1) \simeq \boldsymbol{\psi}_i(k) - \epsilon H_{ii}^{-1} \sum_{j\in\mathcal{N}_i^+} \widehat{\boldsymbol{\eta}}^{(i)}_j \tag{C.8}
$$

$$
\widehat{\boldsymbol{\psi}}^{(i)}_j(k+1) \simeq \begin{cases} \widehat{\boldsymbol{\psi}}^{(i)}_j(k) & \text{if } \gamma^{(i)}_j(k) = 0 \\ \boldsymbol{\psi}_j(k) & \text{if } \gamma^{(i)}_j(k) = 1 \end{cases} \tag{C.9}
$$

$$
\widehat{\boldsymbol{\eta}}^{(i)}_j(k+1) \simeq \begin{cases} \widehat{\boldsymbol{\eta}}^{(i)}_j(k) & \text{if } \gamma^{(i)}_j(k) = 0 \\ H^j_{ij}\boldsymbol{\psi}_j(k) + \sum_{\ell\in\mathcal{N}_j} H^j_{i\ell}\widehat{\boldsymbol{\psi}}^{(i)}_\ell(k) & \text{if } \gamma^{(i)}_j(k) = 1 \end{cases} \tag{C.10}
$$

$$
\widehat{\boldsymbol{\zeta}}^{(i)}_j(k+1) \simeq \begin{cases} \widehat{\boldsymbol{\zeta}}^{(i)}_j(k) & \text{if } \gamma^{(i)}_j(k) = 0 \\ K^j_{ij}\boldsymbol{\psi}_j(k) + \sum_{\ell\in\mathcal{N}_j} K^j_{i\ell}\widehat{\boldsymbol{\psi}}^{(i)}_\ell(k) & \text{if } \gamma^{(i)}_j(k) = 1. \end{cases} \tag{C.11}
$$

where in Eqn. (C.8) we used the fact that $\left.\frac{\partial f^i_1}{\partial \widehat{\boldsymbol{e}}}\right|_{\boldsymbol{x}^*,\widehat{\boldsymbol{\rho}}^*,\widehat{\boldsymbol{e}}^*} = -\epsilon \frac{\partial (f^i_{\boldsymbol{e}})^{-1}}{\partial \widehat{\boldsymbol{e}}} f^i_{\boldsymbol{\rho}}\Big|_{\boldsymbol{x}^*,\widehat{\boldsymbol{\rho}}^*,\widehat{\boldsymbol{e}}^*} = 0$ since $f^i_{\boldsymbol{\rho}}\Big|_{\boldsymbol{x}^*,\widehat{\boldsymbol{\rho}}^*,\widehat{\boldsymbol{e}}} = \nabla_i J(\boldsymbol{x}^*) = \mathbf{0}$, and the fact that $f^i_{\boldsymbol{e}}(\widehat{\boldsymbol{e}}^*) = H_{ii}$. In Eqn.(C.10) we used the fact that $H^j_{i\ell} = \nabla^2_{i\ell} J_j(\boldsymbol{x}^*_j, \{\boldsymbol{x}^*_s\}_{s\in\mathbb{N}_j})$. Finally, in Eqn. (C.11) the matrices $K^j_{i\ell}$ depends on third order derivatives of $J(\boldsymbol{x})$ whose values are unimportant for the analysis of the

stability of the dynamics. By collecting all the variables together, we obtain the system

$$\begin{bmatrix} \boldsymbol{\psi}(k+1) \\ \widehat{\boldsymbol{\psi}}(k+1) \\ \widehat{\boldsymbol{\eta}}(k+1) \\ \widehat{\boldsymbol{\zeta}}(k+1) \end{bmatrix} = \left[\begin{array}{c|ccc} I & 0 & -\epsilon B & 0 \\ \hline C_1(k) & F_1(k) & 0 & 0 \\ C_2(k) & F_2(k) & F_3(k) & 0 \\ C_3(k) & F_4(k) & 0 & F_5(k) \end{array}\right] \begin{bmatrix} \boldsymbol{\psi}(k) \\ \widehat{\boldsymbol{\psi}}(k) \\ \widehat{\boldsymbol{\eta}}(k) \\ \widehat{\boldsymbol{\zeta}}(k) \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\psi}(k+1) \\ \boldsymbol{y}(k+1) \end{bmatrix} = \left[\begin{array}{c|cc} I & 0 \; -\epsilon B \; 0 \\ \hline C(k) & F(k) \end{array}\right] \begin{bmatrix} \boldsymbol{\psi}(k) \\ \boldsymbol{y}(k) \end{bmatrix}. \tag{C.12}$$

where $\boldsymbol{y} = (\widehat{\boldsymbol{\psi}}, \widehat{\xi}, \widehat{\boldsymbol{\zeta}})$ collects the fast dynamic variables. Notice that $F_1(k)$, $F_3(k)$ and $F_5(k)$ are diagonal matrices whose entries are either 1 or 0, depending on the communication between agent success, and, as a consequence, $F(k)$ is a lower triangular matrix, $\forall t$.

Assumption 1 of Lemma C.1.1:
It is finally necessary to prove that the linearized dynamics above satisfies the three assumptions of Lemma C.1.1, where $\boldsymbol{\psi}$ plays the role of $\boldsymbol{x}$ in the Lemma. It is simple to verify by direct inspection that for a fixed $\boldsymbol{\psi}$, the following maps satisfy Assumption 1 of Lemma C.1.1:

$$\widehat{\boldsymbol{\psi}}_j^{(i)} = \boldsymbol{\psi}_j \tag{C.13}$$

$$\widehat{\boldsymbol{\eta}}_j^{(i)} = H_{ij}^j \boldsymbol{\psi}_j + \sum_{s \in \mathcal{N}_j} H_{is}^j \boldsymbol{\psi}_s \tag{C.14}$$

$$\widehat{\boldsymbol{\zeta}}_j^{(i)} = K_{ij}^j \boldsymbol{\psi}_j + \sum_{s \in \mathcal{N}_j} K_{is}^j \boldsymbol{\psi}_s \tag{C.15}$$

in fact, this is equivalent of saying that there exists a matrix G such that $\boldsymbol{y} = G\boldsymbol{\psi}$ satisfies the equality $\boldsymbol{y} = C(k)\boldsymbol{\psi} + F(k)\boldsymbol{y}$ for all $\boldsymbol{\psi}$ and k.

Assumption 2 of Lemma C.1.1:
Consider now the fast dynamics of the system given by the following system:

$$\boldsymbol{z}(k) = F(k-1) \cdots F(0)\boldsymbol{z}(0) = \Omega(k)\boldsymbol{z}(0)$$

Assumptions 2.3.1 and 2.3.2 on the communication among the agents, assure that

$$F_1(T-1)\cdots F_1(0) = \Omega_1(k) = 0$$
$$F_3(T-1)\cdots F_3(0) = \Omega_3(k) = 0$$
$$F_5(T-1)\cdots F_5(0) = \Omega_5(k) = 0$$

in fact when $\gamma_j^{(i)}(k) = 1$, the corresponding rows in the matrices $F_1(k), F_3(k), F_5(k)$ become zero, and this property will be inherited also by the product matrices $\Omega_1(k)$, $\Omega_2(k)$, $\Omega_3(k)$ since all $F_1(k)$, $F_2(k)$, $F_3(k)$ are diagonal. Since all $\gamma_j^{(i)}(k)$ will be equal to one at least once within the window $k \in [0, \cdots, T-1]$, then the matrices $\Omega_1(k)$, $\Omega_2(k)$, $\Omega_3(k)$ must be all zero. Finally, since the matrix $F(k)$ is lower triangular, after a maximum of $(2T+1)$ iterations the product matrix $\Omega(2T+1)$ will be zero and thus $\boldsymbol{z}(2T+1) = 0$. That is, the fast variable dynamic is exponentially stable, since it reaches the equilibrium in a finite number of iteration.

Assumption 3 of Lemma C.1.1:
Finally, consider the slow dynamical system

$$\dot{\boldsymbol{\psi}}(k) = -BG\boldsymbol{\psi}(k). \tag{C.16}$$

which by direct substitution from the previous analysis can be locally written as:

$$\dot{\boldsymbol{\psi}}_i(k) = -H_{ii}^{-1}\left(\sum_{j\in\mathcal{N}_i^+}\left(H_{ij}^j\boldsymbol{\psi}_j + \sum_{s\in\mathcal{N}_i} H_{is}^j\boldsymbol{\psi}_s\right)\right) = -H_{ii}^{-1}H^i\boldsymbol{\psi}$$

where H was defined above and corresponds to the Hessian of the global cost J computed at $\boldsymbol{x}^*$, i.e. $H = \nabla^2 J(\boldsymbol{x}^*)$ and H^i is its i-th block-row,
i.e., $H^i = [\nabla_{i1}^2 J(\boldsymbol{x}^*)\cdots\nabla_{iN}^2 J(\boldsymbol{x}^*)]$. This implies that

$$BG = (\mathrm{diag}(H))^{-1}H\,,$$

therefore, choosing

$$V(\boldsymbol{\psi}) = \frac{1}{2}\boldsymbol{\psi}^\top H\boldsymbol{\psi},$$

as a Lyapunov function, it is straightforward to see that system (C.16) is asymptotically stable since $\dot{V}(\boldsymbol{\psi}(k)) = -\boldsymbol{\psi}^\top(k)H\,(\mathrm{diag}(H))^{-1}H\boldsymbol{\psi}(k) < 0, \boldsymbol{x} \neq 0$ being $H > 0$ by assumption.

Assumption 4 of Lemma C.1.1:

This comes from the observation that the time-variance of the state matrices depends on the specific sequence of packet losses that can occur. Since there are only a finite number of possible different sequences, the assumption is clearly satisfied.

Concluding, system (C.12) satisfies the hypothesis of Lemma C.1.1, and thus there exists $\bar{\epsilon}$, with $0 < \epsilon < \bar{\epsilon}$ such that, by using the resilient block Jacobi Algorithm 4.2,

$$\lim_{t \to \infty} \boldsymbol{x}(k) = \boldsymbol{x}^*.$$

locally exponentially fast.

■

D

Appendix for Chapter 5

D.1 Proof of Proposition 5.5.2 and 5.5.3

Due to (5.12), the eigenvalue $\Phi_i(\epsilon)$ of V, $i = 1, \dots, N$, has the form $\Phi_i(\epsilon) = 1 - 2\frac{\epsilon}{\bar{a}}\gamma_i$, where γ_i is the $i-$th eigenvalue of $L_{\mathcal{G}}$. The convergence rate of dual ascent is given by the second largest eigenvalue of V in modulus (determined either by $\Phi_2(\epsilon)$ or by $\Phi_N(\epsilon)$). The optimal value ϵ^* of ϵ is such that $|\Phi_2(\epsilon^*)| = |\Phi_N(\epsilon^*)|$, from which the value ϵ^* and the corresponding convergence rate immediately follow.
Now, to prove Proposition 5.5.3, note that when $\mathcal{G}$ is a $d-$regular graph, it holds $L_{\mathcal{G}} = (d+1)(I_N - P)$. After expressing the eigenvalues of $L_{\mathcal{G}}$ as a function of the eigenvalues of P, the results for the optimal ϵ and the optimal convergence rate follow.

Regarding the bounds, if $\mu_1 = d > \mu_2 \geq \cdots \geq \mu_N \geq -d$ are the eigenvalues of $A_{\mathcal{G}}^-$, then $\lambda_i = \frac{1}{d+1}(\mu_i + 1)$. For a node i of $\mathcal{G}$, consider $\hat{\mathbf{x}} \in \mathbb{R}^N$ such that $\hat{x}_j = 1$ if $j = i$, $\hat{x}_j = d^{-1}$ if $j|(i,j) \in \mathcal{E}$, and $\hat{x}_j = 0$ otherwise. It holds

$$\mu_N = \min_{\boldsymbol{x} \in \mathbb{R}^N} \frac{\boldsymbol{x}^\top A_{\mathcal{G}}^- \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \leq \frac{\hat{\boldsymbol{x}}^\top A_{\mathcal{G}}^- \hat{\boldsymbol{x}}}{\hat{\boldsymbol{x}}^\top \hat{\boldsymbol{x}}} \leq \frac{-2 + \frac{(d-1)}{d}}{1 + \frac{1}{d}} = -1.$$

Now $\mu_N = (d+1)\lambda_N - 1 \leq -1$ and so $\lambda_N \leq 0$. From $\mu_N \geq -d$ follows $0 \geq \lambda_N \geq -\frac{d-1}{d+1} := \xi$.

Concerning λ_2, for a non complete graph consider nodes i, j such that $(i,j) \notin \mathcal{E}$, and build $\hat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $\hat{x}_j = 1$ if $k = i$,$\hat{x}_k = -1$ if $k = j$ and $\hat{x}_k = 0$ otherwise.
$\mathbf{1}_N$ is the eigenvector related to the eigenvalue d, and it also holds that $\hat{\boldsymbol{x}}^\top \mathbf{1}_N = 0$. Using Rayleigh quotient

$$\mu_2 = \max_{\boldsymbol{x} | \boldsymbol{x}^\top \mathbf{1}_N = 0} \frac{\boldsymbol{x}^\top A_{\bar{\mathcal{G}}} \boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \geq \frac{\hat{\boldsymbol{x}}^\top A_{\bar{\mathcal{G}}} \hat{\boldsymbol{x}}}{\hat{\boldsymbol{x}}^\top \hat{\boldsymbol{x}}} = 0,$$

since there is no edge between node i and j. Therefore $\mu_2 \geq 0$, and so, for λ_2, $\mu_2 = (d+1)\lambda_2 - 1 \geq 0$ and $\lambda_2 \geq \frac{1}{d+1}$. Since $\mu_2 < d$ it follows $\frac{1}{d+1} \leq \lambda_2 \leq 1$.

This part of the proof was obtained starting from Trevisan (2012). As long as $\rho_C \leq \xi$, the smallest value for ρ_{DA} is achieved when $\rho_C = |\lambda_N|$ and λ_2 is as small as possible. In order not to have a bound that depends on d, λ_2 has been set to 0 in this case, and bound (5.15) follows. Instead, when $\rho_C > \xi$, it must hold that $\rho_C = \lambda_2$, and in this case the value of λ_N which minimizes ρ_{DA} is the biggest possible, that is $\lambda_N = 0$. This demonstrates (5.16). In case the graph is complete, both λ_2 and λ_N are 0 (and therefore also ρ_C), and this implies that ρ_{DA} is 0 and the bounds still hold.

D.2 Proof for the matrix form for ADMM

To obtain Formula (5.22), start by fixing the initial condition of λ_{ij} and of z_i to 0. With these initial conditions, expliciting $\sum_{i \in \mathcal{N}_j^+} \lambda_{ij}(k)$ from equation (5.20) and substituting this formula into (5.21), after summing both sides over all $i \in \mathcal{N}_j$, it can be shown that $\sum_{i \in \mathcal{N}_j^+} \lambda_{ij}(k+1) = 0,\ t \geq -1$. Introducing $\bar{w}_j = \sum_{i \in \mathcal{N}_j^+} w_{ij}$, the update of $z_j(k)$ can be rewritten as

$$z_j(k+1) = \frac{1}{\bar{w}_j} \sum_{i \in \mathcal{N}_j^+} w_{ij} x_i(k+1),\ t \geq 0. \tag{D.1}$$

Now, replacing (D.1) in (5.21) and summing in $j \in \mathcal{N}_i^+$

$$\begin{aligned} \sum_{j \in \mathcal{N}_i^+} \lambda_{ij}(k) = & \sum_{j \in \mathcal{N}_i^+} \lambda_{ij}(t-1) + \sum_{j \in \mathcal{N}_i^+} w_{ij} x_i(k) \\ & - \sum_{j \in \mathcal{N}_i^+} w_{ij} \frac{1}{\bar{w}_j} \sum_{s \in \mathcal{N}_j^+} w_{sj} x_s(k),\ t \geq 1. \end{aligned} \tag{D.2}$$

Defining the quantities $\hat{w}_i = \sum_{j\in\mathcal{N}_i^+} w_{ij}$ and $d_i = \frac{\hat{w}_i}{a_i+\hat{w}_i}$, the substitution of (D.1) and (D.2) in (5.19) yields the following

$$x_i(k+1) = (1-d_i)\theta_i + \bar{\lambda}_i(k) + d_i x_i(k) + 2u_i(k), \tag{D.3}$$

$$\bar{\lambda}_i(k) := -\sum_{j\in\mathcal{N}_i^+} \frac{\lambda_{ij}(t-1)}{a_i+\hat{w}_i}, \tag{D.4}$$

$$\begin{aligned} u_i(k) &:= \sum_{j\in\mathcal{N}_i^+} \frac{w_{ij}}{a_i+\hat{w}_i} \sum_{s\in\mathcal{N}_j^+} \frac{w_{sj}}{\bar{w}_j} x_s(k) - \sum_{j\in\mathcal{N}_i^+} \frac{w_{ij}}{a_i+\hat{w}_i} x_i(k) \\ &= \left(\frac{1}{a_i+\hat{w}_i} W_i \operatorname{diag}(\mathbf{1}_N^\top W)^{-1} W^\top - D \right) \boldsymbol{x}(k), \end{aligned}$$

which are valid for $t \geq 1$.
The new variable $\bar{\lambda}_i(k)$ is updated as $\bar{\lambda}_i(k+1) = \bar{\lambda}_i(k) + u_i(k),\ t \geq 1$. Collecting in vector $\bar{\lambda} \in \mathbb{R}^N$ all the $\bar{\lambda}_i$ and using matrices D and U defined in subsection 5.6, the following matrix form for the updating can be obtained, which is valid for all $t \geq 1$:

$$\begin{aligned} \boldsymbol{x}(k+1) &= (I_N - D)\theta + \bar{\lambda}(k) + D\boldsymbol{x}(k) + 2U\boldsymbol{x}(k), \\ \bar{\lambda}(k+1) &= \bar{\lambda}(k) + U\boldsymbol{x}(k). \end{aligned}$$

From the initial conditions $z_i(0) = 0,\ \lambda_{ij}(0) = 0$, the initial conditions for this new algorithm are $\boldsymbol{x}(1) = (I_N - D)\theta$ and $\bar{\lambda}(1) = \mathbf{0}_N$.

Now, adding and subtracting $\boldsymbol{x}(k)$ in the first equation and making further calculation, the update of $\boldsymbol{x}(k)$ can be written in the following way

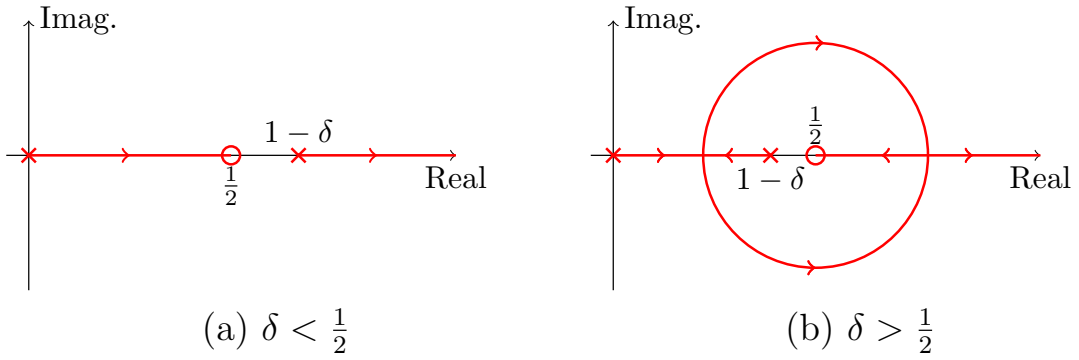
$$\boldsymbol{x}(k+1) = (I_N + D + 2U)\boldsymbol{x}(k) - (D+U)\boldsymbol{x}(t-1),\ t \geq 1,$$

with initial condition $\boldsymbol{x}(0) = 0$ and $\boldsymbol{x}(1) = (I_N - D)\theta$.
From the latter Formula, equation (5.22) follows.

D.3 Proof of Proposition 5.6.2

Due to the symmetry of P and to the form of M and K in Formula (5.23), F is similar to an $N-$blocks diagonal matrix, whose blocks have the following form

$$B_i^F = \begin{bmatrix} 1-\delta+2\delta\lambda_i^2 & -\delta\lambda_i^2 \\ 1 & 0 \end{bmatrix}, \quad i = 1, \ldots, N, \tag{D.5}$$



(a) $\delta < \frac{1}{2}$ (b) $\delta > \frac{1}{2}$

Figure D.1: Root locus for the characteristic polynomial of B_i^F with respect to λ_i, for 2 values of δ

where $\delta = \frac{\mu}{\bar{a}+\mu}$, $0 < \delta < 1$ and λ_i are the eigenvalues of P. The roots of the characteristic polynomial of B_i^F are

$$\xi_{1_i,2_i} = \frac{1-\delta+2\delta\lambda_i^2 \pm \sqrt{\delta^2(1-2\lambda_i^2)^2+1-2\delta}}{2}, \tag{D.6}$$

for all λ_i, $i = 1, \ldots, N$. The eigenvalues related to $\lambda_1 = 1$ can be evaluated from (D.6), and correspond to 1 and δ. To study the eigenvalues of the other blocks, the positive root locus of $z(z-(1-\delta)) + \lambda_i^2(\delta - 2\delta z)$ with respect to the parameter λ_i^2 are studied (note that the polynomial is a rewriting of the characteristic polynomial of B_i^F). The root locus is represented in Figure D.1. In particular, it is necessary to determine the value δ^* for δ which minimizes all $\xi_{1_i,2_i}$.

First consider $\delta = \frac{1}{2}$. In this case the eigenvalues of the $i-$th block can be evaluated by (D.6), and are $\frac{1}{2}$ and λ_i^2.

Denote with $\hat{k}$ the index (equal to 2 or to N) such that $|\lambda_{\hat{k}}| = \rho$. As long as $\lambda_{\hat{k}}^2$ is smaller or equal to $\frac{1}{2}$, the optimal choice for δ is $\frac{1}{2}$. As a matter of fact, for smaller δ, Figure (D.1a) shows that there are eigenvalues bigger than $\frac{1}{2}$, while for bigger δ, the eigenvalue in δ of block 1 is bigger than $\frac{1}{2}$.

When $\lambda_{\hat{k}}^2$ is bigger than $\frac{1}{2}$, the biggest eigenvalue in absolute value is determined by $\lambda_{\hat{k}}$, as can be inferred when $\delta < 2$, and from the same when $\delta > 1/2$ together with the fact that the modulus of the eigenvalues when $\lambda_i^2 = \frac{1}{2}$ is bigger than the modulus when $\lambda_i^2 = 0$. In order to minimize $\max\left\{|\xi_{1_{\hat{k}}}|, |\xi_{2_{\hat{k}}}|\right\}$ it suffices to choose a δ such that the term

under root is 0. The calculations give the following values for δ^* and for $|\xi_{1_{\hat{k}}}| = |\xi_{2_{\hat{k}}}|$:

$$\delta^* = \frac{1}{1 + 2\sqrt{\rho_C^2 - \rho_C^4}}, \qquad |\xi_{1_{\hat{k}}}| = |\xi_{2_{\hat{k}}}| = \frac{\rho_C}{\rho_C + \sqrt{1 - \rho_C^2}}.$$

It can be verified that $|\xi_{1_{\hat{k}}}| > \delta^*$, so when $\lambda_{\hat{k}}^2 > \frac{1}{2}$ the optimal convergence rate is $\rho_{ADMM_P} = |\xi_{1_{\hat{k}}}|$.

D.4 Proof of Theorem 5.9.4

The proof of Theorem 5.9.4 is based on the theory of ergodic coefficients for positive matrices Seneta (2006), applied to the particular case of stochastic matrices. To proceed with the proof, first the algorithm iteration is written in a matrix form:

Matrix form for ra-AC First introduce the indicator variables $\chi_i(k)$ and $\chi_{(i,j)}(k)$, $i, j \in \{1, \dots, N\}$. The variable $\chi_i(k)$ is equal to 1 if node i wakes up at time k, otherwise is 0; at this regard, recall that since a broadcast asymmetric protocol is adopted only one node turns on at each iteration. Concerning $\chi_{(i,j)}(k)$, the variable is 1 if node i wakes up at time k, if $j \in \mathcal{N}_i^{\text{out}}$ and if the edge $(i, j) \in \mathcal{E}$ is reliable at time k, while it is 0 otherwise. Formally

$$\chi_i(k) = \begin{cases} 1 & \text{if node } i \text{ wakes up at time } k \\ 0 & \text{otherwise} \end{cases} \tag{D.7}$$

and

$$\chi_{(i,j)}(k) = \begin{cases} 1 & \text{if } \chi_i(k) = 1, (i,j) \in \mathcal{E} \text{ active at time } k \\ 0 & \text{otherwise}^{1} \end{cases} \tag{D.8}$$

Observe that $\chi_{(i,j)}(k)$ is considered identically 0 for all k, if $(i, j) \notin \mathcal{E}$ and that $\sum_{i=1}^{N} \chi_i(k) = 1$. In the following, only the matrices which describe the evolution of variable $\boldsymbol{q}(k)$ are described, since the same matrices drive the evolution of variable $\boldsymbol{s}(k)$. Using the indicator variables the update for the total sent-mass counter $\sigma_{i,q}(k)$ and for the total received-mass counter $\rho_{j,q}^{(i)}(k)$ can be rewritten as

$$\sigma_{i,q}(k+1) = \sigma_{i,q}(k) + \chi_i(k) \frac{q_i(k)}{|\mathcal{N}_i^{\text{out}}| + 1} \tag{D.9}$$

$$\rho_{j,q}^{(i)}(k+1) = \rho_{j,q}^{(i)}(k) - \chi_i(k)\chi_{(i,j)}(k) \left(\rho_{j,q}^{(i)}(k) - \sigma_{i,q}(k+1) \right) \tag{D.10}$$

Introduce now the variables

$$\nu_{j,q}^{(i)}(k) = \sigma_{i,q}(k) - \rho_{j,q}^{(i)}(k), \ \forall (i,j) \in \mathcal{E}.$$

These variables indicate how much of the mass sent by node i is still to be received by node j. If at time $k-1$ node i turns on and the communication between node i and j (where j is a neighbour of i) is successful, then $\nu_{j,q}^{(i)}(k)$ is 0, otherwise it contains the information missing in node j. Using equations (D.9) and (D.10) the update of these variables can be written as

$$\begin{aligned}\nu_{j,q}^{(i)}(k+1) &= \Big[1-\chi_i(k)\chi_{(i,j)}(k)\Big]\Big[\sigma_{i,q}(k+1)-\rho_{j,q}^{(i)}(k)\Big] \\ &= \Big[1-\chi_i(k)\chi_{(i,j)}(k)\Big]\left[\chi_i(k)\frac{q_i(k)}{|\mathcal{N}_i^{\text{out}}|+1}+\nu_{j,q}^{(i)}(k)\right].\end{aligned} \tag{D.11}$$

These variables are now exploited to rewrite the update of vector $\boldsymbol{q}(k)$ in a matrix form. Note that these quantities are not actually computed by the nodes, and are just auxiliary variables used to enable the matrix version of the update.

To rewrite the update for the $q_i(k)$ variable, consider three different cases:

- if $\chi_i(k) = 1$, it holds $q_i(k+1) = \frac{q_i(k)}{|\mathcal{N}_i^{\text{out}}|+1}$;
- if $\chi_j(k) = 1$ and $i \in \mathcal{N}_j^{\text{in}}$ and $\chi_{(j,i)}(k) = 1$, then

$$q_i(k+1) = \sigma_{j,q}(k+1) - \rho_{i,q}^{(j)}(k) + q_i(k) = \nu_{i,q}^{(j)}(k) + \frac{q_j(k)}{|\mathcal{N}_j^{\text{out}}|+1} + q_i(k);$$

- if $\chi_j(k) = 1$ and $i \in \mathcal{N}_j^{\text{in}}$ and $\chi_{(j,i)}(k) = 0$ or if $i \notin \mathcal{N}_j^{\text{in}}$, then it holds

$$q_i(k+1) = q_i(k).$$

The above three cases are all captured by the following update

$$\begin{aligned}q_i(k+1) &= \chi_i(k)\frac{q_i(k)}{|\mathcal{N}_i^{\text{out}}|+1} + [1-\chi_i(k)]\cdot \\ &\cdot\left\{\sum_{j\neq i}\left[\chi_j(k)\chi_{(j,i)}(k)\left(\nu_{i,q}^{(j)}(k)+\frac{q_j(k)}{|\mathcal{N}_j^{\text{out}}|+1}\right)\right]+q_i(k)\right\}.\end{aligned} \tag{D.12}$$

Now introduce the column vector $\boldsymbol{\nu}_q(k) = [\nu_{j,q}^{(i)}(k)] \in \mathbb{R}^E$, which collects all different

$\nu_{j,q}^{(i)}(k)$. Moreover define the row vector

$$\boldsymbol{q}_a(k) = [\boldsymbol{q}(k)^\top \ \boldsymbol{\nu}_q(k)^\top] \in \mathbb{R}^{N+E}.$$

The aim is to find matrix $M(k) \in \mathbb{R}^{(N+E)\times(N+E)}$ according to which it holds

$$\boldsymbol{q}_a(k+1) = \boldsymbol{q}_a(k)M(k). \tag{D.13}$$

Start by considering the $i-$th row of matrix $M(k)$, with $i \in \{1,\dots,N\}$. The element $[M(k)]_{ii}$ indicates how $q_i(k)$ influences $q_i(k+1)$, so

$$[M(k)]_{ii} = \frac{\chi_i(k)}{|\mathcal{N}_i^{\text{out}}|+1} + [1-\chi_i(k)].$$

The element $[M(k)]_{ij},\ j \in \{1,\dots,N\}\setminus\{i\}$ indicates how $q_i(k)$ influences $q_j(k+1)$. It holds

$$[M(k)]_{ij} = [1-\chi_j(k)]\left[\frac{\chi_i(k)\chi_{(i,j)}(k)}{|\mathcal{N}_i^{\text{out}}|+1}\right].$$

Finally, if $\ell \in \{N+1,\dots,N+E\}$ is such that $[\boldsymbol{q}_a(k)]_\ell = \nu_{j,q}^{(r)}(k)$, the element $[M(k)]_{i\ell}$ indicates how $q_i(k)$ influences $\nu_{j,q}^{(r)}(k)$. It holds

$$[M(k)]_{i\ell} = \begin{cases} \left[1-\chi_i(k)\chi_{(i,j)}(k)\right]\left[\frac{\chi_i(k)}{|\mathcal{N}_i^{\text{out}}|+1}\right] & \text{if } r=i \\ 0 & \text{if } r\neq i \end{cases}$$

Now analyze the $h-$th row of $M(k)$, with $h \in \{N+1,\dots,N+E\}$. Suppose that $[\boldsymbol{q}_a(k)]_h = \nu_{\ell,q}^{(r)}(k)$. Reasoning as before, it holds

$$\begin{aligned}[M(k)]_{hh} &= 1-\chi_r(k)\chi_{(r,\ell)}(k),\\ [M(k)]_{h\ell} &= [1-\chi_\ell(k)]\left[\chi_r(k)\chi_{(r,\ell)}(k)\right].\end{aligned}$$

and all the other elements in the $h-$th row are 0.
Using the matrices $M(k)$ just defined and introducing variables $\nu_{j,s}^{(i)}(k) = \sigma_{i,s}(k) - \rho_{j,s}^{(i)}(k),\ \forall (i,j)\in\mathcal{E}$ and $\boldsymbol{s}_a(k) = [\boldsymbol{s}(k)^\top \ \boldsymbol{\nu}_s(k)^\top]$, the evolution of $\boldsymbol{q}_a(k)$ and $\boldsymbol{s}_a(k)$ is given by

$$\begin{cases} \boldsymbol{q}_a(k+1) = \boldsymbol{q}_a(k)M(k) \\ \boldsymbol{s}_a(k+1) = \boldsymbol{s}_a(k)M(k) \end{cases} \tag{D.14}$$

Recall that the first N elements of vectors $\boldsymbol{q}_a(k)$ and $\boldsymbol{s}_a(k)$ corresponds respectively to $\boldsymbol{q}(k)$ and $\boldsymbol{s}(k)$.

In the following, the properties of the matrices that describe the algorithm are studied.

Properties of matrices $M(k)$ Introducing the set $\mathcal{M}$, which collects all possible matrices $M(k)$, the following lemma holds true.

Lemma D.4.1. *The set of matrices $\mathcal{M}$ satisfies*

1. *$\mathcal{M}$ is a finite set;*
2. *each $M \in \mathcal{M}$ is a row-stochastic matrix;*
3. *each positive element in any matrix $M \in \mathcal{M}$ is lower bounded by a positive constant c;*
4. *given $\tau_{\max} = Nh\tau$, for all $k \geq 0$, the stochastic matrix*

$$V^{(\tau_{\max})}(k) = M(k)M(k+1)\cdots M(k+\tau_{\max}-1),\ M(k) \in \mathcal{M},$$

is such that its first N columns have all the elements which are strictly positive.

Proof. (1) Each matrix $M \in \mathcal{M}$ depends on which node wakes up and on which communication links from this node to its neighbours work. Since the number of all possible combinations is finite (and in particular equal to $\sum_{i=1}^{N} 2^{|\mathcal{N}_i^{\text{out}}|}$) the property is verified. (2) Consider first the $i-$th row of M, with $i \in \{1,\dots,N\}$. Then, either $\chi_i(k) = 0$, from which it follows

$$\begin{cases} [M(k)]_{ii} = 1 \\ [M(k)]_{ij} = 0 \text{ if } j \in \{1,\dots,N\} \setminus \{i\} \\ [M(k)]_{ij} = 0 \text{ if } j \in \{N+1,\dots,N+E\} \end{cases},$$

or $\chi_i(k) = 1$ (and all other $\chi_j(k) = 0, j \neq i$), implying

$$\begin{cases} [M(k)]_{ii} = \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} \\ [M(k)]_{ij} = \frac{\chi_{(i,j)}(k)}{|\mathcal{N}_i^{\text{out}}|+1} \qquad \text{if } j \in \{1,\dots,N\} \setminus \{i\} \end{cases}$$

and

$$[M(k)]_{i\ell} = \frac{1-\chi_{(i,j)}(k)}{|\mathcal{N}_i^{\text{out}}|+1}$$

for those $\ell \in \{N+1,\dots,N+E\}$ for which there exists a $j \in \{1,\dots,N\} \setminus \{i\}$ such that $\psi_\ell(k) = \nu_{j,q}^{(i)}(k)$ and $[M(k)]_{i\ell} = 0$ otherwise. Note that in both cases the sum of the row is 1.

Consider now the $h-$th row of matrix $M(k)$, with h such that $[\boldsymbol{q}_a(k)]_h = \nu_{\ell,q}^{(r)}(k)$. If $\chi_r(k) = 0$ it holds

$$\begin{cases} [M(k)]_{hh} = 1 \\ [M(k)]_{h\ell} = 0 \text{ if } \ell \in \{1, \dots, N+E\} \setminus \{i\} \end{cases}.$$

On the other hand, if $\chi_r(k) = 1$

$$\begin{cases} [M(k)]_{hh} = 1 - \chi_{(r,\ell)}(k) \\ [M(k)]_{h\ell} = \chi_{(r,\ell)}(k) \\ [M(k)]_{hj} = 0 \text{ if } j \in \{1, \dots, N+E\} \setminus \{i, \ell\} \end{cases}$$

In both cases the row sums up to 1.
(3) This directly follows from the construction of $M(k)$.
(4) Define $V^{(h)}(k) = M(k)M(k+1)\dots M(k+h-1), k \geq 0, h \geq 1$, $V^{(0)}(k) = I_N, k \geq 0$, which can be divided as

$$V^{(h)}(k) = \begin{bmatrix} A_{11}^{(h)}(k) & A_{12}^{(h)}(k) \\ A_{21}^{(h)}(k) & A_{22}^{(h)}(k) \end{bmatrix},$$

with $A_{11}^{(h)}(k) \in \mathbb{R}^{N\times N}, A_{22}^{(h)}(k) \in \mathbb{R}^{E\times E}$, $A_{12}^{(h)}(k) \in \mathbb{R}^{N\times E}$ and $A_{21}^{(h)}(k) \in \mathbb{R}^{E\times N}$. Since every matrix $M \in \mathcal{M}$ is such that $[M]_{ii} > 0$ if $i \in \{1, \dots, N\}$, it holds, for $h \geq 1$, that if in the product which yields $V^{(h)}(k)$ there exists a matrix with the element in position (i, j) strictly greater than 0, then also $[V^{(h)}(k)]_{ij} > 0$. Due to Assumptions 2.3.2 and 2.3.1, after $h\tau$ iterations, all the links in graph $\mathcal{G}$ have successfully transmitted at least once. Moreover, if at time $k + \Delta$, $0 \leq \Delta \leq h\tau$, the communication link of the edge $(i, j) \in \mathcal{E}$ is reliable, then considering index s such that $[\boldsymbol{q}_a(\cdot)]_s = \nu_{j,q}^{(i)}(\cdot)$, it holds $[M(k + \Delta)]_{sj} > 0$. As a consequence, each row of $A_{21}^{(h\tau)}(k)$ has at least one non zero element. Using a similar reasoning, for all $(i, j) \in \mathcal{E}$, it holds that $[V^{(h\tau)}(k)]_{ij} > 0$. Since graph $\mathcal{G}$ is connected, it holds that all the elements of $A_{11}^{((N-1)h\tau)}(k)$ are strictly positive. Due to the last two properties, choosing $\tau_{\max} = Nh\tau$, matrix $V^{(\tau_{\max})}(k)$ has the first N columns with all the elements strictly positive. ∎

Remark D.4.2. The constant $\tau_{\max}$ has been evaluated in the worst possible scenario. As a matter of fact it has been evaluated assuming that in graph $\mathcal{G}$ there are at least two nodes that communicate with each other in no less than $N - 1$ steps and that the communication along one link fails $h - 1$ times consecutively. This implies that in a random network $\mathcal{G}$, where the diameter of the graph is usually much smaller than the

number of nodes, the actual constant $\tau_{\max}$ according to which the first N columns of $V^{(\tau_{\max})}(k)$ are strictly positive will be, in general, much smaller.

It is now time to use ergodicity theory and to prove the convergence of the algorithm.

Ergodicity theory and convergence of ra-AC First some useful concepts of ergodicity theory to be later applied to prove the convergence of the algorithm are recalled. An exhaustive explanation for ergodicity theory can be found in Seneta (2006). Given a stochastic matrix $P \in \mathbb{R}^{N\times N}$, a coefficient of ergodicity for P quantifies how much its rows are different from each other . Two well-known coefficients of ergodicity for a stochastic matrix P are

$$\delta(P) := \max_{j} \max_{i_1,i_2} |[P]_{i_1 j} - [P]_{i_2 j}|\,,$$
$$\lambda(P) := 1 - \min_{i_1,i_2} \sum_{j} \min\{[P]_{i_1 j}, [P]_{i_2 j}\}\,.$$

These coefficients are proper (that is $\delta(P) = 0$ and $\lambda(P) = 0$ if and only if $P = \mathbf{1}_n \boldsymbol{w}^\top$, with $\boldsymbol{w}$ such that $\boldsymbol{w}^\top \mathbf{1}_n = 1$), and, as all the coefficients of ergodicity, $0 \le \delta(P) \le 1$ and $0 \le \lambda(P) \le 1$.
Consider now a stochastic matrix P such that $\delta(P) < \psi$. Selecting two elements in any column of P, the difference between these two elements is necessarily smaller than ψ. Consider now a vector $\boldsymbol{q} \in \mathbb{R}^N$ which sums to 0, that is $\mathbf{1}_N^\top \boldsymbol{q} = 0$. Define the related quantities

$$q_{\text{pos}} = \sum_{i|y_i>0} q_i \ge 1, \quad q_{\text{neg}} = \sum_{i|y_i<0} q_i \le 0, \quad q_{\text{pos}} + q_{\text{neg}} = 0,$$

and also, for any $j \in \{1, \dots, N\}$, the quantities

$$\overline{P}_j = \max_s [P]_{sj}, \quad \underline{P}_j = \min_s [P]_{sj}, \quad \overline{P}_j - \psi \le \underline{P}_j \le \overline{P}_j.$$

Suppose now that[2] $q_{\text{pos}} > 0$. The aim is to find an upper and lower bound for $[\boldsymbol{q}^\top P]_j$, for any $1 \le j \le N$. The maximum value for this product is achieved in case the positive elements of vector $\boldsymbol{q}$ are multiplied by $\overline{P}_j$ and the negative elements of the same vector are multiplied by $\underline{P}_j$, that is

$$[\boldsymbol{q}^\top P]_j \le q_{\text{pos}}\overline{P}_j + q_{\text{neg}}\underline{P}_j \le q_{\text{pos}}\overline{P}_j + q_{\text{neg}}\overline{P}_j - q_{\text{neg}}\psi$$

which reduces to $[\boldsymbol{q}^\top P]_j \le -q_{\text{neg}}\psi$. The minimum value of $[\boldsymbol{q}^\top P]_j$ is instead produced if

[2] It is possible to verify that the bound in Equation D.15, is still verified if $q_i = 0\ \forall i$.

the negative elements are multiplied by $\overline{P}_j$ and the positive ones by $\underline{P}_j$, i.e.

$$[\boldsymbol{q}^\top P]_j \geq q_{\text{neg}}\overline{P}_j + q_{\text{pos}}\underline{P}_j \leq q_{\text{neg}}\overline{P}_j + q_{\text{pos}}\overline{P}_j - q_{\text{pos}}\psi$$

which implies $[\boldsymbol{q}^\top P]_j \geq -q_{\text{pos}}\psi$. From these two bounds the next bound follows

$$\left|[\boldsymbol{q}^\top P]_j\right| \leq \psi \sum_{i=1}^{N} |q_i| \tag{D.15}$$

This bound will be used to prove the convergence of the algorithm. In particular, the stochastic matrix involved will be the forward product of the matrices that define the evolution of the algorithm as seen in (D.14), that is matrix $T(k) = M(0)M(1)\cdots M(k)$. This matrix allows to evaluate $\boldsymbol{q}_a(k+1)$ given $\boldsymbol{q}_a(0)$ (supposing the initial time is 0). The next property is important to evaluate the coefficient $\delta(T(k))$. The property holds for $\delta(\cdot)$ and $\lambda(\cdot)$ when the product of row stochastic matrices is considered: given r stochastic matrices $P_1, \ldots, P_r$, then

$$\delta(P_1 P_2 \cdots P_r) \leq \prod_{i=1}^{r} \lambda(P_i). \tag{D.16}$$

As a consequence if some of the matrices P_i are such that $\lambda(P_i) < 1$, then also $\delta(P_1 P_2 \cdots P_r)$ will be strictly less than 1. A stochastic matrix P such that $\lambda(P) < 1$ is called scrambling, and a sufficient condition for P to be scrambling is that at least one column is strictly positive,as can be verified by the definition of $\lambda(\cdot)$.
It is possible to apply this theory to the forward product of matrices. Start by defining the following

$$W(r) = \prod_{k=(r-1)\tau_{\max}}^{r\tau_{\max}-1} M(k),\ r \geq 1,\ M(k) \in \mathcal{M}$$

which, by Lemma D.4.1, have strictly positive columns. As a consequence, $\lambda(W(r)) < 1$ for all $r \geq 1$. Moreover, the number of different $W(r)$ is finite since matrices $M(k)$ are finite and the Assumptions 2.3.2 and 2.3.1 have to be satisfied. Collecting all $W(r)$ in set $\mathcal{W}$, it is possible to define value

$$d = \max_{W \in \mathcal{W}} \lambda(W),$$

which is strictly smaller than 1.
The following lemma holds

Lemma D.4.3. *The constant $\beta = d^{1/(2\tau_{\max})}$, $0 < \beta < 1$, is such that $\delta(T(k)) \leq \beta^k$ for*

$k \geq \tau_{\max}$.

Proof. If $k \geq \tau_{\max}$, $T(k)$ can be rewritten as

$$T(k) = W(1) \cdots W(r) M(s\tau_{\max}) M(r\tau_{\max}+1) \cdots M(r\tau_{\max}+\Delta)$$

with $r = \lfloor k/\tau_{\max} \rfloor$ and $\Delta = k - r\tau_{\max}$, $0 \leq \Delta \leq \tau_{\max} - 1$. As a consequence, using Formula (D.16)

$$\delta(T(k)) \leq \lambda(W(1)) \cdots \lambda(W(r)) \lambda \left(\prod_{j=0}^{\Delta} M(r\tau_{\max}+j) \right) \leq d^r.$$

Since $r \geq k/(2\tau_{\max})$, $d^r \leq d^{k/(2\tau_{\max})}$, so choosing $\beta = d^{1/(2\tau_{\max})}$, $\delta(T(k)) \leq \beta^k$. ∎

Lemma D.4.3 implies that the coefficient of ergodicity for $T(k)$ converges to 0 as k goes to infinity.

Before showing the convergence of ra-AC, it is necessary to show that each component of $\boldsymbol{s}(k)$ is lower bounded by a constant $\mu > 0$, since the variable $\boldsymbol{x}(k)$ is obtained through the Hadamard division by $\boldsymbol{s}(k)$. Note that if $k \geq \tau_{\max}$, the elements of the first N columns of $T(k)$ are strictly bigger than $c^{\tau_{\max}}$. As a consequence, $s_i(k+1) \geq c^{\tau_{\max}} \sum_{j=1}^{N} s_j(0) \geq c^{\tau_{\max}}$. On the other hand, since the first N elements of the diagonals of matrices $M(k)$ are strictly positive, if $1 \leq k \leq \tau_{\max} - 1$, then $s_i(k+1) \geq c^k s_i(0) \geq c^{\tau_{\max}}$, since $0 < c < 1$ and $\boldsymbol{s}(0) = \mathbf{1}_N$. Therefore it is possible to choose μ as $c^{\tau_{\max}}$.

Finally, it is possible to prove convergence: first the exponential convergence is shown in case vector $\boldsymbol{v}$ is zero mean, $\bar{v} = 0$, and then this will be generalized to the case, $\bar{v} \neq 0$. For $k \geq \tau_{\max}$, by Lemma D.4.3 it holds $\delta(T(k)) \leq \beta^k$, where $T(k)$ is such that $\boldsymbol{q}_a(k+1) = \boldsymbol{q}_a(0)T(k)$. Starting from Formula (D.15), and remembering that the first N elements of $\boldsymbol{q}_a(0)$ are $\boldsymbol{q}(0)$ and the other elements are 0, for all $1 \leq i \leq N$ it holds

$$|[\boldsymbol{q}_a(k+1)]_i| = |[\boldsymbol{q}_a(0)T(k+1)]_i| \leq \beta^k \sum_i |q_i|$$

Now, since for $1 \leq i \leq N$, $q_i(k+1) = [\boldsymbol{q}_a(k+1)]_i$ and the elements of $\boldsymbol{s}(k)$ are strictly greater than μ,

$$|q_i(k+1)| = \left| \frac{s_i(k+1)}{s_i(k+1)} q_i(k+1) \right| \leq \beta^k \sum_i |q_i|$$

which implies

$$|x_i(k+1)| = \left| \frac{q_i(k+1)}{s_i(k+1)} \right| \leq \frac{1}{s_i(k)} \beta^k \sum_i |q_i| \leq \frac{1}{\mu} \beta^k \sum_i |q_i|$$

and then

$$\|\boldsymbol{x}(k+1)\|^2 \leq \frac{N}{\mu^2\beta^2}(\beta^2)^{k+1}\left(\sum_i |q_i|\right)^2 \leq \frac{N^2}{\mu^2\beta^2}(\beta^2)^{k+1}\|\boldsymbol{x}(0)\|^2$$

where the last inequality is a consequence of the Cauchy-Schwarz inequality and the fact that $\boldsymbol{x}(0) = \boldsymbol{q}(0)$.

Defining $C_\infty := \frac{N^2}{\mu^2\beta^2}$ the latter becomes

$$\|\boldsymbol{x}(k)\|^2 \leq C_\infty (d^{1/\tau_{\max}})^k \|\boldsymbol{x}(0)\|^2, \quad k \geq \tau_{\max} + 1.$$

If $0 \leq k \leq \tau_{\max}, T(k)$ is still stochastic and it surely holds that $\delta(T(k)) \leq 1$, so applying a similar reasoning

$$\|\boldsymbol{x}(k)\|^2 \leq C_k (d^{1/\tau_{\max}})^k \|\boldsymbol{x}(0)\|^2, \quad C_k = \frac{N^2}{\mu^2 (d^{1/\tau_{\max}})^k}.$$

Introducing $C = \max\{C_1, \ldots, C_{\tau_{\max}}, C_\infty\} = C_{\tau_{\max}}$ it holds

$$\|\boldsymbol{x}(k)\|^2 \leq C (d^{1/\tau_{\max}})^k \|\boldsymbol{x}(0)\|^2, \quad k \geq 0, \tag{D.17}$$

that is the algorithm exponentially converges when vector $\boldsymbol{v}$ is 0, since the mean $\bar{v}$ is 0, and the vector $\boldsymbol{x}(k)$ is converging to $0\mathbf{1}_N$.

It is now possible to finally generalize to the case in which $\boldsymbol{v}$ is such that $\bar{v} \neq 0$. Introducing vector $\boldsymbol{v}_0 = \boldsymbol{v} - \bar{v}\mathbf{1}_N$, consider two evolutions of the algorithm, one initialized using $\boldsymbol{v}_0$ and the other initialized to $\boldsymbol{v}$. At each time step k the same matrix $M(k)$ is applied for both initializations. The subscript 0 is used to indicate the variables of the evolution starting from the zero-mean vector $\boldsymbol{v}_0$ and the subscript $\bar{v}$ to indicate those starting from vector $\boldsymbol{v}$ (vectors $\boldsymbol{s}(k)$ and $\boldsymbol{s}_a(k)$ do not have a subscript since they are the same in both evolutions). Remembering that $\boldsymbol{s}(0) = \mathbf{1}_N$,

$$\boldsymbol{x}_0(0) = \frac{\boldsymbol{q}_0(0)}{\boldsymbol{s}(0)}, \quad \boldsymbol{x}_{\bar{v}} = \frac{\boldsymbol{q}_{\bar{v}}(0)}{\boldsymbol{s}(0)} = \frac{\boldsymbol{q}_0(0) + \bar{v}\mathbf{1}_N}{\boldsymbol{s}(0)} = \frac{\boldsymbol{q}_0(0)}{\boldsymbol{s}(0)} + \bar{v}\mathbf{1}_N$$

so $\boldsymbol{x}_{\bar{v}}(0) = \boldsymbol{x}_0(0) + \bar{v}\mathbf{1}_N$. Moreover, it is possible to verify that $\boldsymbol{\phi}_{\bar{v}}^{(q)}(0) = \boldsymbol{\phi}_0^{(q)}(0) + \bar{v}\boldsymbol{s}_a(0)$,

according to which for all $1 \leq i \leq N$

$$\begin{aligned}
[\boldsymbol{x}_0(k)]_i &= \left[\frac{\boldsymbol{\phi}_0^{(q)}(k)}{\boldsymbol{s}_a(k)}\right]_i = \left[\frac{\boldsymbol{\phi}_0^{(q)}(0)T(k-1)}{\boldsymbol{s}_a(0)T(k-1)}\right]_i \\
[\boldsymbol{x}_{\bar{v}}(k)]_i &= \left[\frac{\boldsymbol{\phi}_{\bar{v}}^{(q)}(k)}{\boldsymbol{s}_a(k)}\right]_i = \left[\frac{\boldsymbol{\phi}_{\bar{v}}^{(q)}(0)T(k-1)}{\boldsymbol{s}_a(0)T(k-1)}\right]_i \\
&= \left[\frac{\boldsymbol{\phi}_0^{(q)}(0)T(k-1)}{\boldsymbol{s}_a(0)T(k-1)}\right]_i + \left[\bar{v}\frac{\boldsymbol{s}_a(0)T(k-1)}{\boldsymbol{s}_a(0)T(k-1)}\right]_i \\
&= [\boldsymbol{x}_0(k)]_i + \bar{v}.
\end{aligned}$$

This proves that $\boldsymbol{x}_{\bar{v}}(k)$ can be always obtained as $\boldsymbol{x}_0(k) + \bar{v}\mathbf{1}_N$ for all $k \geq 0$, and therefore, since for $\boldsymbol{x}_0(k)$ Equation (D.17) holds, and so

$$\|\boldsymbol{x}_{\bar{v}}(k) - \bar{v}\mathbf{1}_N\|^2 \leq C(d^{1/\tau_{\max}})^k \|\boldsymbol{x}_{\bar{v}}(0) - \bar{v}\mathbf{1}_N\|, \quad k \geq 0.$$

This implies that the exponential convergence of $\boldsymbol{x}$ to $\bar{v}\mathbf{1}_N$ holds for any vector $\boldsymbol{v} \in \mathbb{R}^N$.

E

Appendix for Chapter 6

E.1 Proof of Proposition 6.5.1

The following proof has strong similarity to the proof for the ra-AC algorithm. The main (apparent) difference is that in the ra-AC algorithm the matrix form for the update of $\boldsymbol{q}_a()$ is given by the multiplication of a row-vector with a row-stochastic matrix, while here the matrix form for the update of $\boldsymbol{y}_a$ and $\boldsymbol{z}_a$ is given by the multiplication of a column-stochastic matrix with a column-vector. Obviously, to move from one representation to the other it is enough to transpose the update.

Start by observing that, only nodes in $\tilde{N}_i(k) \cup \{i\}$ update the variables x, g^{old}, g, h^{old}, h. Moreover, observe that the matrix $S(k)$ can be seen as a *selection* matrix which selects the nodes in $\tilde{N}_i(k) \cup \{i\}$. This explains the vector form of the first five equations in (6.7).

Now, to each edge (i,j), $j \in \mathcal{N}_i^{\text{out}}$, an indicator function variable $X_{i,j}(k)$ is associated as follows:

$$X_{i,j}(k) = \begin{cases} 1, & \text{if } (i,j) \text{ reliable at time } k \\ 0, & \text{if } (i,j) \text{ not reliable at time } k. \end{cases}$$

For the sake of simplicity, only the update of $\boldsymbol{y}_a$ is considered (since the update of $\boldsymbol{z}_a$ is

similar). Recall that

$$y_i(k+1) = \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1)\right). \tag{E.1}$$

Observe that, for $j \in \mathcal{N}_i^{\text{out}}$, by using the indicator function defined above, it holds that

$$\rho_{j,y}^{(i)}(k+1) = X_{i,j}(k)\sigma_{i,y}(k+1) + (1 - X_{i,j}(k))\,\rho_{j,y}^{(i)}(k).$$

Since

$$\nu_{j,y}^{(i)}(k) = \sigma_{i,y}(k) - \rho_{j,y}^{(i)}(k),$$

and

$$\sigma_{i,y}(k+1) = \sigma_{i,y}(k) + y_i(k+1),$$

it follows that

$$\nu_{j,y}^{(i)}(k+1) = (1 - X_{i,j}(k))\left(\nu_{j,y}^{(i)}(k) + \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1)\right]\right) \tag{E.2}$$

and that

$$y_j(k+1) = y_j(k) + X_{i,j}(k)\left[y_i(k+1) + g_j(k+1) - g_j^{\text{old}}(k+1) + \nu_{j,y}^{(i)}(k)\right]$$

and, in turn, that

$$\begin{aligned} y_j(k+1) = y_j(k) &+ X_{i,j}(k)\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}y_i(k) + \\ &+ X_{i,j}(k)\left[g_j(k+1) - g_j^{\text{old}}(k+1) + \nu_{j,y}^{(i)}(k) + \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[g_i(k+1) - g_i^{\text{old}}(k+1)\right]\right]. \end{aligned} \tag{E.3}$$

From (E.1) and (E.3) it is possible to write that

$$\begin{aligned} \boldsymbol{y}(k+1) = &\left[\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(e_i + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j\right)e_i^T + \sum_{h\neq i} e_h e_h^T\right]\boldsymbol{y}(k) + \sum_{j\in\tilde{\mathcal{N}}_i} e_j e_{(i,j)}^T\,\boldsymbol{\nu} + \\ &+ \left[\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(e_i + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j\right)e_i^T + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j e_j^T\right]\left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1)\right) \\ &= M_{VV}(k)\boldsymbol{y} + M_{V\mathcal{E}}(k)\,\boldsymbol{\nu}_y(k) + T_V(k)\left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1)\right). \end{aligned}$$

From (E.2), it follows

$$\begin{aligned}\boldsymbol{\nu}_y(k+1) = &\left[\sum_{j\in\bar{\mathcal{N}}_i} e_{(i,j)}e_{(i,j)}^T + \sum_{(r,s)\,:\,r\neq i} e_{(r,s)}e_{(r,s)}^T\right]\boldsymbol{\nu}_y + \\ &+\left[\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\sum_{j\in\bar{\mathcal{N}}_i} e_{(i,j)}e_i^T\right]\left(\boldsymbol{y}+\boldsymbol{g}(k+1)-\boldsymbol{g}^{\text{old}}(k+1)\right) \\ &= M_{\mathcal{E}V}(k)\boldsymbol{y}(k) + M_{\mathcal{E}\mathcal{E}}(k)\boldsymbol{\nu}_y(k) + T_{\mathcal{E}}(k)\left(\boldsymbol{g}(k+1)-\boldsymbol{g}^{\text{old}}(k+1)\right).\end{aligned}$$

The above computations explain the vector-form illustrated in equations (6.7).

The fact that $M(k)$ is a column-stochastic matrix can be shown by verifying that the sum of the elements of each column is equal to one. In particular, note that all the columns of $M(k)$ have only one element equal to 1, except for the column relative to node i, and that the sum of the latter is 1.

E.2 Proof of Lemma 6.5.3

The proof is only the first equality; the second one can be proved analogously. The proof is by induction. The property is trivially true for $k=0$. Indeed, according to the Initialization block, $y_\ell(0) = g_\ell(0) = g_\ell^{\text{old}}(0) = \nu_{j,y}^{(\ell)}(0) = 0$ for all ℓ and $j \in \mathcal{N}_\ell^{\text{out}}$; the fact that $g_\ell(0) = g_\ell^{\text{old}}(0) = 0$ implies that also $g_\ell^{\text{old}}(1) = 0$ for all ℓ. Now, assume the property to be true for k. The idea is to show that it holds also for $k+1$. Without loss of generality, assume that node i is activated at iteration k. Then,

$$\begin{aligned}&\sum_{\ell=1}^{N}\left(y_\ell(k+1) + \sum_{j\in\mathcal{N}_\ell^{\text{out}}}\nu_{j,y}^{(\ell)}(k+1)\right) = \mathbf{1}^T\boldsymbol{y}_a(k+1) \\ &= \mathbf{1}^T M(k)\boldsymbol{y}_a(k) + \mathbf{1}^T T(k)\left(\boldsymbol{g}(k+1)-\boldsymbol{g}^{\text{old}}(k+1)\right) \\ &= \sum_{\ell=1}^{N}\left(y_\ell(k) + \sum_{j\in\mathcal{N}_\ell^{\text{out}}}\nu_{j,y}^{(\ell)}(k)\right) + g_i(k+1) - g_i^{\text{old}}(k+1) + \\ &\qquad + \sum_{j\in\mathcal{N}_i^{\text{out}}} X_{i,j}(k)\left(g_j(k+1)-g_j^{\text{old}}(k+1)\right) \\ &= \sum_{\ell=1}^{N} g_\ell(k) + \sum_{j\,\in\,\tilde{\mathcal{N}}_i(k)\cup\{i\}}\left(g_j(k+1)-g_j^{\text{old}}(k+1)\right),\end{aligned}$$

which follows from the properties

$$\mathbf{1}^T M(k) = \mathbf{1}^T, \qquad \mathbf{1}^T T(k) = e_i^T + \sum_{j \in \tilde{\mathcal{N}}_i} e_j^T,$$

and the inductive hypothesis

$$\sum_{\ell=1}^{N} \left(y_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{\text{out}}} \nu_{j,y}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} g_\ell(k).$$

By simple algebraic manipulations, it holds

$$\begin{aligned}
&\sum_{\ell=1}^{N} g_\ell(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \Big(g_j(k+1) - g_j^{\text{old}}(k+1) \Big) \\
&= \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \sum_{j \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \Big(g_j(k+1) - g_j^{\text{old}}(k+1) \Big) \\
&= \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k+1) + \sum_{j \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \Big(g_j(k) - g_j^{\text{old}}(k+1) \Big).
\end{aligned}$$

Now, observe that, if $\ell \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}$ then $g_\ell(k+1) = g_\ell(k)$, and, if $\ell \in \tilde{\mathcal{N}}_i(k) \cup \{i\}$ then $g_\ell^{\text{old}}(k+1) = g_\ell(k)$. Then, from the previous expression, it follows

$$\sum_{\ell=1}^{N} g_\ell(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \Big(g_j(k+1) - g_j^{\text{old}}(k+1) \Big) = \sum_{\ell=1}^{N} g_\ell(k+1).$$

This concludes the proof.

E.3 General results on discrete-time nonlinear systems

The proofs and results of this appendix can be found in Section 6 of the technical report Bof, Carli, and Schenato (2017e). Consider the system

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon \phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \end{cases} \tag{E.4}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_1}$, $\boldsymbol{\xi} \in \mathbb{R}^{n_2}$, $\phi : \mathbb{N} \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_1}$, $\varphi : \mathbb{N} \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$, $\epsilon > 0$ and with given initial conditions $\boldsymbol{x}(0)$, $\boldsymbol{\xi}(0)$.

For a given $\bar{k} \in \mathbb{N}$, consider the system, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}_{\bar{k}}(k+1) = \varphi\left(k, \boldsymbol{x}(\bar{k}), \tilde{\boldsymbol{\xi}}_{\bar{k}}(k)\right), \tag{E.5}$$

initialized by $\boldsymbol{\xi}_{\bar{k}}(\bar{k}) = \boldsymbol{\xi}(\bar{k})$, where $\boldsymbol{\xi}(\bar{k})$ is obtained ruling system (E.4) up to $\bar{k}$. Given $\bar{k}$, assume that, for $k \geq \bar{k}$, there exists a sequence

$$k \rightarrow \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k), \tag{E.6}$$

in general dependent on $\boldsymbol{x}(\bar{k})$ and $\boldsymbol{\xi}(\bar{k})$, such that the evolution

$$\tilde{\boldsymbol{\xi}}_{\bar{k}}(k) := \boldsymbol{\xi}_{\bar{k}}(k) - \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k) \tag{E.7}$$

satisfies the property

$$\|\tilde{\boldsymbol{\xi}}_{\bar{k}}(k)\| \leq C_{\bar{k}}\, \rho_{\bar{k}}^{k-\bar{k}}\, \|\tilde{\boldsymbol{\xi}}_{\bar{k}}(\bar{k})\|, \tag{E.8}$$

for suitable $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$, that is $\tilde{\boldsymbol{\xi}}'_{\bar{k}} = \mathbf{0}$ is an exponentially stable point for the evolution in (E.7). Basically, the property in (E.8) establishes that there exists a trajectory $\boldsymbol{\xi}^*$ to which the trajectory of the variable $\boldsymbol{\xi}$, generated keeping the variable $\boldsymbol{x}$ constant, converges asymptotically.

Next, assume that, for each k, the variable $\boldsymbol{\xi}$ has already reached the asymptotic convergence to the corresponding trajectory $\boldsymbol{\xi}^*$. More precisely, observe that there exists a family of sequences of the type (E.6), where each sequence starts from a different index k. From this family it is possible to build the following new sequence

$$k \rightarrow \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k), \tag{E.9}$$

where, to the index k, the first element of the sequence which starts at k is associated. Based on (E.9), consider the system

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon\, \phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right). \tag{E.10}$$

Assume that $\boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)$ is such that there exists a suitable map $\tilde{\phi} : \mathbb{N} \times \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_1}$ such that (E.10) can be, equivalently, rewritten as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon\, \tilde{\phi}\left(k, \boldsymbol{x}(k)\right), \tag{E.11}$$

that is, $\tilde{\phi}\left(k, \boldsymbol{x}(k)\right) = \phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right)$. We make the following assumption.

Assumption E.3.1. Let $\boldsymbol{x}^*$ be an equilibrium point for (E.11). There exists $r > 0$ such

that $\tilde{\phi}$ is continuously differentiable on $D = \{\boldsymbol{x} \in \mathbb{R}^{n_1} \mid \|\boldsymbol{x} - \boldsymbol{x}^*\| < r\}$ and the Jacobian matrix $[\partial\tilde{\phi}/\partial\boldsymbol{x}]$ is bounded and Lipschitz on D, uniformly in k. In addition, defining

$$A(k) = I + \epsilon \frac{\partial \tilde{\phi}}{\partial \boldsymbol{x}}(k; \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*},$$

and considering the auxiliary system

$$\tilde{\boldsymbol{x}}(k+1) = A(k)\tilde{\boldsymbol{x}}(k), \tag{E.12}$$

$\tilde{\boldsymbol{x}} = 0$ is an exponentially stable equilibrium point for (E.12).

The following Proposition characterizes the convergence properties of system (E.4).

Proposition E.3.2. *Consider system in (E.4). For any $\bar{k}$, assume that there exists a sequence as in (E.6) such that property (E.8) is satisfied. Consider system in (E.10). Let $\boldsymbol{x}^*$ be an equilibrium point for (E.10). Assume Assumption (E.3.1) holds true. Then, there exist $r > 0$ and $\epsilon^* > 0$, such that, for all $\epsilon \in (0, \epsilon^*]$ and for all $x(0) \in B_r^n = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{x}^*\| < r\}$, the trajectory $\boldsymbol{x}(t)$ generated by (E.4), converges exponentially to $\boldsymbol{x}^*$, i.e., there exist $C > 0$ and $0 < \lambda < 1$ such that*

$$\|\boldsymbol{x}(k) - \boldsymbol{x}^*\| \leq C\lambda^k \|\boldsymbol{x}(0) - \boldsymbol{x}^*\|.$$

References

Acevedo J. J., Arrue B. C., Diaz-Bañez J. M., Ventura I., Maza I., and Ollero A. One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots. *Journal of Intelligent & Robotic Systems*, 74 (1-2):269–285, 2014.

Acevedo J. J., Arrue B. C., Maza I., and Ollero A. Cooperative perimeter surveillance with a team of mobile robots under communication constraints. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5067–5072. IEEE, 2013.

Aghajan H. and Cavallaro A. *Multi-camera networks: principles and applications*. Academic Press, Cambridge, MA, 2009.

Alberton R., Carli R., Cenedese A., and Schenato L. Multi-agent perimeter patrolling subject to mobility constraints. In *Proceedings of the 2012 American Control Conference*, pages 4498–4503, 2012.

Albino V., Berardi U., and Dangelico R. M. Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of Urban Technology*, 22(1):3–21, 2015.

Anderson B. D. and Moore J. B. *Linear optimal control*, volume 197. Prentice-Hall, Upper Saddle River, NJ, 1971.

Argaez M., Ramirez C., and Sanchez R. An ℓ_1-algorithm for underdetermined systems and applications. In *Proceedings of the 2011 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 1–6. IEEE, 2011.

Aysal T. C., Yildiz M. E., Sarwate A. D., and Scaglione A. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, July 2009. ISSN 1053-587X.

Balch O. Smart cities: how to build sustainable urban environments? *Forbes*, Mar. 20, 2013.

Barooah P. and Hespanha J. Estimation on graphs from relative measurements: Distributed algorithms and fundamental limit. *IEEE Control Systems Magazine*, 27 (4):57–74, 2007.

Belgioioso G., Cenedese A., and Michieletto G. Distributed partitioning strategies with visual optimization for camera network perimeter patrolling. In *Proceedings of the 55th IEEE Conference on Decision and Control, 2016*, pages 5912–5917. IEEE, 2016.

Bénézit F., Blondel V., Thiran P., Tsitsiklis J., and Vetterli M. Weighted gossip: distributed averaging using non-doubly stochastic matrices. In *Proceedings of the 2010 IEEE International Symposium on Information Theory Proceedings*, pages 1753–1757. IEEE, 2010.

Bertsekas D. P. *Constrained optimization and Lagrange multiplier methods*. Academic Press, Cambridge, MA, 2014.

Bertsekas D. P. and Tsitsiklis J. N. *Parallel and distributed computation: numerical methods*, volume 23. Prentice-Hall, Upper Saddle River, NJ, 1989.

Bertsimas D. and Tsitsiklis J. N. *Introduction to linear optimization*, volume 6. Athena Scientific, Belmont, MA, 1997.

Bianchi P., Hachem W., and Iutzeler F. A stochastic coordinate descent primal-dual algorithm and applications to distributed optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2016.

Bianchi P., Hachem W., and Iutzeler F. A stochastic coordinate descent primal-dual algorithm and applications to large-scale composite optimization. *arXiv preprint arXiv:1407.0898*, 2014.

Bin S.-Y. and Lin C.-H. An implementable distributed state estimator and distributed bad data processing schemes for electric power systems. *IEEE Transactions on Power Systems*, 9(3):1277–1284, 1994.

Blondel V. D., Hendrickx J. M., Olshevsky A., and Tsitsiklis J. N. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, 2005*, pages 2996–3000. IEEE, 2005.

Bof N., Carli R., and Schenato L. Average consensus with asynchronous updates and unreliable communication. In *Proceedings of the 20th IFAC World Congress, 2017*, volume 50, pages 601–606. IFAC, 2017a.

Bof N., Todescato M., Carli R., and Schenato L. Robust distributed estimation for localization in lossy sensor networks. In *Proceedings of the 6th IFAC Workshop on Distributed Estimation and control in Networked System, 2016*, pages 250–255. IFAC, 2016a.

Bof N., Carli R., Cenedese A., and Schenato L. Asynchronous distributed camera network patrolling under unreliable communication. *IEEE Transactions on Automatic Control*, 62(11):5982–5989, 2017b.

Bof N., Carli R., Notarstefano G., Schenato L., and Varagnolo D. Newton-Raphson Consensus under lossy communication for peer-to-peer optimization. *arXiv preprint arXiv:1707.09178*, 2017c.

Bof N., Carli R., and Schenato L. On the performance of consensus based versus Lagrangian based algorithms for quadratic cost functions. In *Proceedings of the 2016 European Control Conference*, pages 160–165. IEEE, 2016b.

Bof N., Carli R., and Schenato L. Is ADMM always faster than average consensus? *Provisionally accepted on Automatica*, 2017d.

Bof N., Carli R., and Schenato L. Lyapunov theory for discrete time systems. Technical report, 2017e. [Online] Available at http://automatica.dei.unipd.it/tl_files/utenti2/bof/Papers/NoteDiscreteLyapunov.pdf.

Bolognani S., Carli R., and Todescato M. State estimation in power distribution networks with poorly synchronized measurements. In *Proceedings of the 53rd IEEE Conference on Decision and Control, 2014*, pages 2579–2584. IEEE, 2014.

Bolognani S., Favero S. D., Schenato L., and Varagnolo D. Consensus-based distributed sensor calibration and least-square parameter identification in WSNs. *International Journal of Robust and Nonlinear Control*, 20(2):176–193, 2010.

Borra D., Pasqualetti F., and Bullo F. Continuous graph partitioning for camera network surveillance. *Automatica*, 52:227–231, 2015.

Boyd S., Ghosh A., Prabhakar B., and Shah D. Randomized gossip algorithms. *IEEE Transactions on Information Theory/ACM Transactions on Networking*, 52(6): 2508–2530, June 2006.

Boyd S., Diaconis P., and Xiao L. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.

Boyd S., Parikh N., Chu E., Peleato B., and Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Boyd S. and Vandenberghe L. *Convex optimization*. Cambridge University Press, Cambridge, GB, 2004.

Bryant C. Europe's manufacturers experiment with the smart factory. *Financial Times*, Apr. 10, 2014.

Bullo F., Carli R., and Frasca P. Gossip coverage control for robotic networks: dynamical systems on the space of partitions. *SIAM Journal on Control and Optimization*, 50(1):419–447, 2012.

Cardwell D. Grid sensors could ease disruptions of power. *The New York Times*, Feb. 05, 2015.

Cardwell D. Smart bus can alter its route to avoid jams. *The Times*, May 10, 2017.

Carli R., Fagnani F., Speranzon A., and Zampieri S. Communication constraints in the average consensus problem. *Automatica*, 44(3):671–684, 2008.

Carli R., Notarstefano G., Schenato L., and Varagnolo D. Distributed quadratic programming under asynchronous and lossy communications via Newton-Raphson Consensus. In *Proceedings of the 2015 IEEE European Control Conference*, 2015.

Carron A., Todescato M., Carli R., Schenato L., and Pillonetto G. Multi-agents adaptive estimation and coverage control using gaussian regression. In *Proceedings of the 2015 IEEE European Control Conference*, 2015.

Carron A., Todescato M., Carli R., and Schenato L. An asynchronous consensus-based algorithm for estimation from noisy relative measurements. *IEEE Transactions on Control of Network Systems*, 1(3):283–295, 2014.

Cassandras C., Lin X., and Ding X. An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Transactions on Automatic Control*, 58(4): 947–961, 2013.

Cattivelli F. S. and Sayed A. H. Diffusion strategies for distributed Kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, 55(9):2069–2084, 2010.

Chang T.-H., Hong M., Liao W.-C., and Wang X. Asynchronous distributed ADMM for large-scale optimization- part I: algorithm and convergence analysis. *IEEE Transactions on Signal Processing*, 64(12):3118–3130, 2016.

Chazal M., Jouini E., and Tahraoui R. Production planning and inventories optimization: a backward approach in the convex storage cost case. *Journal of Mathematical Economics*, 44(9):997–1023, 2008.

Chen Y., Tron R., Terzis A., and Vidal R. Corrective consensus: converging to the exact average. In *Proceedings of the 49th IEEE Conference on Decision and Control, 2010*, pages 1221–1228. IEEE, 2010.

Chevaleyre Y. Theoretical analysis of the multi-agent patrolling problem. *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302 – 308, sep. 2004.

Chu H.-N., Glad A., Simonin O., Sempé F., Drogoul A., and Charpillet F. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, 2007*, volume 1, pages 442–449. IEEE, 2007.

Colias M. and Higgins T. GM to test fleet of self-driving cars in New York. *New York Times*, Oct. 17, 2017.

Conejo A. J., de la Torre S., and Canas M. An optimization approach to multiarea state estimation. *IEEE Transactions on Power Systems*, 1(22):213–221, 2007.

Cook D. J., Youngblood M., and Das S. K. A multi-agent approach to controlling a smart environment. *Designing smart homes*, 4008:165–182, 2006.

Curtis S. Humans could be banned from driving within the next 25 years as autonomous cars take over the roads. *New York Times*, Oct. 9, 2017.

Czyzowicz J., Gasieniec L., Kosowski A., and Kranakis E. Boundary patrolling by mobile agents with distinct maximal speeds. In *Algorithms–ESA 2011*, pages 701–712. Springer, 2011.

Di Lorenzo P. and Scutari G. Next: in-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Ding C., Song B., Morye A., Farrell J. A., and Roy-Chowdhury A. K. Collaborative sensing in a distributed ptz camera network. *Image Processing, IEEE Transactions on*, 21(7):3282–3295, 2012.

Dobre C. and Xhafa F. Parallel programming paradigms and frameworks in big data era. *International Journal of Parallel Programming*, 42(5):710–738, 2014.

Dominguez-Garcia A. D. and Hadjicostis C. N. Coordination and control of distributed energy resources for provision of ancillary services. In *Proceedings of the 1st IEEE International Conference on Smart Grid Communications, 2010*, pages 537–542. IEEE, 2010.

Domínguez-García A. D. and Hadjicostis C. N. Distributed strategies for average consensus in directed graphs. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, 2011*, pages 2124–2129. IEEE, 2011.

Dominguez-Garcia A. D., Hadjicostis C. N., and Vaidya N. H. Distributed algorithms for consensus and coordination in the presence of packet-dropping communication links-part I: Statistical moments analysis approach. *arXiv preprint arXiv:1109.6391*, 2011.

Dresner K. and Stone P. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

Eisen M., Mokhtari A., and Ribeiro A. Decentralized quasi-newton methods. *arXiv preprint arXiv:1605.00933*, 2016.

Erseghe T. A distributed and scalable processing method based upon ADMM. *IEEE Signal Processing Letters*, 19(9):563–566, 2012.

Erseghe T., Zennaro D., Dall'Anese E., and Vangelista L. Fast consensus by the Alternating Direction Multipliers Method. *IEEE Transactions on Signal Processing*, 59(11):5523–5537, 2011.

Esterle L. and Grosu R. Cyber-physical systems: challenge of the 21st century. *e & i Elektrotechnik und Informationstechnik*, 133(7):299–303, 2016.

Estrin D., Girod L., Pottie G., and Srivastava M. Instrumenting the world with wireless sensor networks. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2033–2036. IEEE, 2001.

Franceschetti M. and Meester R. *Random networks for communication: from statistical physics to information systems*, volume 24. Cambridge University Press, Cambridge, GB, 2008.

Garin F. and Schenato L. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked Control Systems*, pages 75–107. Springer, London, GB, 2010.

Ghadimi E., Teixeira A., Shames I., and Johansson M. Optimal parameter selection for the Alternating Direction Method of Multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015. ISSN 0018-9286.

Ghadimi E., Shames I., and Johansson M. Accelerated gradient methods for networked optimization. *arXiv preprint arXiv:1211.2132*, 2012.

Gharesifard B. and Cortes J. Distributed continuous-time convex optimization on weight-balanced digraphs. *IEEE Transactions on Automatic Control*, 59(3):781–786, 2014.

Giridhar A. and Kumar P. Distributed clock synchronization over wireless networks: algorithms and analysis. In *Proceedings of the 45th IEEE Conference on Decision and Control, 2006*, pages 4915–4920. IEEE, 2006.

Horn R. A. and Johnson C. R. *Matrix Analysis*. Cambridge University Press, Cambridge, GB, 1985.

Horst R., Pardalos P. M., and Van Thoai N. *Introduction to global optimization*. Springer Science & Business Media, Berlin/Heidelberg, D, 2000.

Horton C. In Taiwan, modest test of driverless bus may hint at big things to come. *New York Times*, Sep. 28, 2017.

Hou I.-H. and Kumar P. Real-time communication over unreliable wireless links: a theory and its applications. *IEEE Wireless Communications*, 19(1), 2012.

Huck S. M., Kariotoglou N., Summers S., Raimondo D. M., and Lygeros J. Design of importance-map based randomized patrolling strategies. In *Complexity in Engineering (COMPENG), 2012*, pages 1–6. IEEE, 2012.

Iocchi L., Nardi D., and Salerno M. Reactivity and deliberation: a survey on multi-robot systems. In *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 9–32. Springer, 2000.

Ipakchi A. and Albuyeh F. Grid of the future. *IEEE Power and Energy Magazine*, 7 (2):52–62, 2009.

Iutzeler F., Bianchi P., Ciblat P., and Hachem W. Asynchronous distributed optimization using a randomized Alternating Direction Method of Multipliers. In

Proceedings of the 52nd IEEE Conference on Decision and Control, 2013, pages 3671–3676. IEEE, 2013.

Iutzeler F., Bianchi P., Ciblat P., and Hachem W. Explicit convergence rate of a distributed Alternating Direction Method of Multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2016.

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

Kar S. and Moura J. M. Distributed consensus algorithms in sensor networks with imperfect communication: link failures and channel noise. *IEEE Transactions on Signal Processing*, 57(1):355–369, 2009.

Kar S. and Moura J. M. Distributed consensus algorithms in sensor networks: quantized data and random link failures. *IEEE Transactions on Signal Processing*, 58 (3):1383–1400, 2010.

Kariotoglou N., Raimondo D. M., Summers S. J., and Lygeros J. Multi-agent autonomous surveillance: a framework based on stochastic reachability and hierarchical task allocation. *Journal of dynamic systems, measurement, and control*, 137(3):031008, 2015.

Kawamura A. and Kobayashi Y. Fence patrolling by mobile agents with distinct speeds. *Distributed Computing*, 28(2):147–154, 2015.

Kekatos V. and Giannakis G. B. Distributed robust power system state estimation. *IEEE Transactions on Power Systems*, 28(2):1617–1626, 2013.

Khalil H. K. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, third edition, 2001.

Kia S. S., Cortés J., and Martínez S. Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, 55: 254–264, 2015.

Kim K.-D. and Kumar P. R. Cyber–physical systems: a perspective at the centennial. *Proceedings of the IEEE*, 100(Special Centennial Issue):1287–1308, 2012.

Kotkin J. The world's smartest cities. *Forbes*, Mar. 12, 2009.

Kung H. *Synchronized and asynchronous parallel algorithms for multiprocessors*. Carnegie-Mellon University, 1976.

Lee S. and Nedić A. Asynchronous gossip-based random projection algorithms over networks. *IEEE Transactions on Automatic Control*, 61(4):953–968, 2016.

Lin P., Ren W., and Song Y. Distributed multi-agent optimization subject to nonidentical constraints and communication delays. *Automatica*, 65:120–131, 2016.

Ling Q., Shi W., Wu G., and Ribeiro A. Dlm: Decentralized linearized alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(15): 4051–4064, 2015.

Liu J. and Morse A. S. Accelerated linear iterations for distributed averaging. *Annual Reviews in Control*, 35(2):160–165, 2011.

Makhdoumi A. and Ozdaglar A. Convergence rate of distributed ADMM over networks. *arXiv preprint arXiv:1601.00194*, 2016.

Mansoori F. and Wei E. Superlinearly convergent asynchronous distributed network Newton method. *arXiv preprint arXiv:1705.03952*, 2017.

Mao T. and Ray L. Frequency-based patrolling with heterogeneous agents and limited communication. *arXiv preprint arXiv:1402.1757*, 2014.

Marelli D. E. and Fu M. Distributed weighted least-squares estimation with fast convergence for large-scale systems. *Automatica*, 51:27–39, 2015.

Marr B. What everyone must know about industry 4.0. *Forbes*, Jun. 20, 2016a.

Marr B. Why everyone must get ready for the 4th industrial revolution. *Forbes*, Apr. 5, 2016b.

Moreau L. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

Muthukrishnan S., Ghosh B., and Schultz M. H. First-and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of computing systems*, 31(4):331–354, 1998.

National Science Foundation . Cyber-Physical Systems (CPS) NSF 08-611. http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm, 2008. Accessed: 2017-10-12.

Nedić A. Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, June 2011.

Nedić A. and Ozdaglar A. *Convex optimization in signal processing and communications*, chapter Cooperative Distributed Multi-Agent Optimization, pages 340–386. Cambridge University Press, Cambridge, GB, 2010.

Nedić A. and Liu J. A Lyapunov approach to discrete-time linear consensus. In *Proceedings of the 2014 IEEE Global Conference on Signal and Information Processing*, pages 842–846. IEEE, 2014.

Nedić A. and Olshevsky A. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2015.

Nedić A., Olshevsky A., and Shi W. Achieving geometric convergence for distributed optimization over time-varying graphs. *arXiv preprint arXiv:1607.03218*, 2016.

Nedić A. and Ozdaglar A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Nedić A., Ozdaglar A., and Parrilo P. A. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

Nishihara R., Lessart L., Recht B., Packard A., and Jordan M. I. A general analysis of the convergence of ADMM. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Notarnicola I. and Notarstefano G. Asynchronous distributed optimization via randomized dual proximal gradient. *IEEE Transactions on Automatic Control*, 2016.

Notarstefano G. and Bullo F. Distributed abstract optimization via constraints consensus: theory and applications. *IEEE Transactions on Automatic Control*, 56(10): 2247–2261, October 2011.

O'Boyle M. How a smart grid relies on customer demand response to manage wind and solar. *Forbes*, Mar. 13, 2017.

Olshevsky A. and Tsitsiklis J. N. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, 2009.

Oreshkin B. N., Coates M. J., and Rabbat M. G. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, 2010.

Palomar D. P. and Chiang M. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8): 1439–1451, 2006.

Pasqualetti F., Carli R., and Bullo F. Distributed estimation via iterative projections with application to power network monitoring. *Automatica*, 48(5):747–758, 2012.

Pasqualetti F., Zanella F., Peters J. R., Spindler M., Carli R., and Bullo F. Camera network coordination for intruder detection. *IEEE Transactions on Control Systems Technology*, 22(5):1669–1683, 2014.

Patterson S., Bamieh B., and El Abbadi A. Distributed average consensus with stochastic communication failures. In *Proceedings of the 46th IEEE Conference on Decision and Control, 2007*, pages 4215–4220. IEEE, 2007.

Peng Z., Xu Y., Yan M., and Yin W. ARock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38 (5):A2851–A2879, 2016.

Perlroth N. Smart city technology may be vulnerable to hackers. *The New York Times*, Apr. 21, 2015.

Poole S. The fourth industrial revolution review: adapt to new technology or perish. *The Guardian*, Jan. 6, 2017.

Portugal D. and Rocha R. A survey on multi-robot patrolling algorithms. *Technological innovation for sustainability*, pages 139–146, 2011.

Puccinelli D. and Haenggi M. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and systems magazine*, 5(3):19–31, 2005.

Qu G. and Li N. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 2017.

Quain J. R. Cars will talk to one another. Exactly how is less certain. *The New York Times*, Mar. 09, 2017.

Quinn E. L. Privacy and the new energy infrastructure. *SSRN Electronic Journal*, 2009.

Raimondo D. M., Kariotoglou N., Summers S., and Lygeros J. Probabilistic certification of pan-tilt-zoom camera surveillance systems. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, 2011*, pages 2064–2069. IEEE, 2011.

Ravazzi C., Frasca P., Ishii H., and Tempo R. A distributed randomized algorithm for relative localization in sensor networks. In *Proceedings of the 2013 European Control Conference*, pages 1776–1781. IEEE, 2013.

Rogers A., Ramchurn S. D., and Jennings N. R. Delivering the smart grid: challenges for autonomous agents and multi-agent systems research. In *AAAI*, 2012.

Sahai A. and Grover P. Demystifying the Witsenhausen counterexample [ask the experts]. *IEEE Control Systems*, 30(6):20–24, 2010.

Sanburn J. How smart traffic lights could transform your commute. *Time*, May 05, 2015.

Seneta E. *Non-negative matrices and Markov chains*. Springer Science & Business Media, Berlin/Heidelberg, D, 2006.

Sherson T., Heusdens R., and Kleijn W. B. Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory. *arXiv preprint arXiv:1706.02654*, 2017.

Shi W., Ling Q., Yuan K., Wu G., and Yin W. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

Simonetto A. and Leus G. Distributed asynchronous time-varying constrained optimization. In *Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers, 2014*, pages 2142–2146. IEEE, 2014.

Singh S. Smart cities - a $1.5 trillion market opportunity. *Forbes*, Jun. 19, 2014.

Song B., Kamal A. T., Soto C., Ding C., Farrell J. A., and Roy-Chowdhury A. K. Tracking and activity recognition through consensus in distributed camera networks. *IEEE Transactions on Image Processing*, 19(10):2564–2579, 2010.

Sou K. C., Weimer J., Sandberg H., and Johansson K. H. Scheduling smart home appliances using mixed integer linear programming. In *Proceedings of the 50th*

IEEE Conference on Decision and Control and European Control Conference, 2011, pages 5144–5149. IEEE, 2011.

Teixeira A., Ghadimi E., Shames I., Sandberg H., and Johansson M. Optimal scaling of the ADMM algorithm for distributed quadratic programming. In *Proceedings of the 52nd IEEE Conference on Decision and Control and European Control Conference, 2013*, pages 6868–6873. IEEE, 2013.

Teixeira A., Ghadimi E., Shames I., Sandberg H., and Johansson M. The ADMM algorithm for distributed quadratic problems: parameter selection and constraint preconditioning. *IEEE Transactions on Signal Processing*, 64(2):290–305, 2016.

Todescato M., Cavraro G., Carli R., and Schenato L. A robust block-jacobi algorithm for quadratic programming under lossy communications. *IFAC-PapersOnLine*, 48(22):126–131, 2015.

Todescato M., Bof N., Cavraro G., Carli R., and Schenato L. Generalized gradient optimization over lossy networks for partition-based estimation. *arXiv preprint, arXiv:1710.10829*.

Totty M. The rise of the smart city. *The Wall Street Journal*, Apr. 16, 2017.

Trevisan L. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6): 1769–1786, 2012.

Tsai C.-F., Lin W.-C., and Ke S.-W. Big data mining with parallel computing: a comparison of distributed and mapreduce methodologies. *Journal of Systems and Software*, 122:83–92, 2016.

Tsianos K. I., Lawlor S., and Rabbat M. G. Push-sum distributed dual averaging for convex optimization. In *Proceedings of the 51st IEEE Conference on Decision and Control, 2012*, pages 5453–5458. IEEE, 2012.

Tsitsiklis J., Bertsekas D., and Athans M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

Tsitsiklis J. N. Problems in decentralized decision making and computation. Technical report, Massachusetts Instituite of Technology Cambridge Lab For Information and Decision Systems, 1984.

Twentyman J. Smart buildings could open door to hackers. *Financial Times*, May 15, 2017.

Vaidya N. H., Hadjicostis C. N., and Dominguez-Garcia A. D. Distributed algorithms for consensus and coordination in the presence of packet-dropping communication links-part II: coefficients of ergodicity analysis approach. *arXiv preprint arXiv:1109.6392*, 2011.

Varagnolo D., Zanella F., Cenedese A., Gianluigi P., and Schenato L. Newton-Raphson Consensus for distributed convex optimization. *IEEE Transactions on Automatic Control*, 61(4):994 – 1009, 2016.

Walsh B. Miami's smart grid: A blueprint for the power future. *Time*, Apr. 22, 2009.

Wei E. and Ozdaglar A. On the O(1/k) convergence of asynchronous distributed Alternating Direction Method of Multipliers. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*, pages 551–554, 2013.

Wei E., Ozdaglar A., and Jadbabaie A. A distributed Newton method for network utility maximization - Part I: algorithm. *IEEE Transactions on Automatic Control*, 58 (9):2162–2175, 2013a.

Wei E., Ozdaglar A., and Jadbabaie A. A distributed Newton method for network utility maximization - Part II: convergence. *IEEE Transactions on Automatic Control*, 58(9):2176 – 2188, 2013b.

Wheeland M. How Florida and Colorado are trying to build smart cities from the ground up. *The Guardian*, Aug. 9, 2016.

Witsenhausen H. S. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):131–147, 1968.

Xiao L., Boyd S., and Lall S. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, 2005*, pages 63–70. IEEE, 2005.

Yang B. and Johansson M. Distributed optimization and games: a tutorial overview. *Networked Control Systems*, pages 109–148, 2011.

Zamalloa M. Z. and Krishnamachari B. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 3(2):7, 2007.

Zanella F., Varagnolo D., Cenedese A., Pillonetto G., and Schenato L. Newton-Raphson consensus for distributed convex optimization. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, 2011*, pages 5917–5922. IEEE, 2011.

Zanella F., Varagnolo D., Cenedese A., Pillonetto G., and Schenato L. Asynchronous Newton-Raphson Consensus for distributed convex optimization. In *Proceedings of the 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems, 2012*. IEEE, 2012.

Zargham M., Ribeiro A., Ozdaglar A., and Jadbabaie A. Accelerated dual descent for network flow optimization. *IEEE Transactions on Automatic Control*, 59 (4):905–920, 2014.

Zhang R. and Kwok J. Asynchronous distributed ADMM for consensus optimization. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1701–1709, 2014.

Acknowledgments

...and finally another step in my travel turns to the conclusion. In these three years, many people have sustained and helped me, and above all showed their faith in me. I especially appreciate all the patience accorded to me when I was living those periods in which my recurring thought was "What am I doing? Am I really fit to it?". Even though my reactions to your encouragements were probably not really gratifying for you (given my situation and doubts), without all your support these periods would have been extremely difficult to overcome. Some of you will receive special thanks in the following. However, know that if you feel like I neglected you, you should blame my very short memory, not my heart. In the latter there is a place for each and every one of you.

The first thanks are for my supervisor, Prof. Luca Schenato. You really helped me, guiding me and giving me very useful suggestions in the course of all my Ph.D.. You also encouraged me a lot, both during these three years and also when I decided to become "the prodigal daughter", leaving *optimization* to try my luck with *cooking*. Thank you very much for everything.

Together with Luca, it would be impossible for me not to thank my co-supervisor, Prof. Ruggero Carli. Again, the support you gave me not only concerned the research (with all the time spent on intricate formulas), but also my future intentions. I really appreciate it.

Now it is my parent's turn, together with my whole family: I am really lucky to have such a crazy family. I really love all of you, you are always there when I really need you. A special mention is for my mother: you are a true example of love, patience, sensibility and perseverance.

Big thanks also go to my office mates (both former and current). I can vividly remember many times in which you patiently listened to my worries and helped me, and also many times in which you encouraged me to at least try new things. I also appreciated your cake/cookies enthusiasm! Thank you for all the funny moments, the serious ones and also the stressful ones. We lived these three years in a very full way, we were more than just colleagues! Among you, special thanks are for Giacomo. We helped

and sustained each other a lot, not only on Ph.D.'s matters. This is what really matters, and I really appreciate it!

Even though my office mates received a major mention, I want to extend my thanks to all the people I met in the university world, both here in Padua and in Stanford, with special thanks to my Stanford's supervisor, Prof. Marco Pavone, who kindly welcomed me in his lab for my six months permanence there, and to Prof. Ettore Fornasini, who mentored me in my first months of the Ph.D.. I learned a lot from all of you and I had the possibility to engage in deep and interesting discussions. A note also for the reviewers of this thesis, Angelia Nedić and Mikael Johansson, who carefully read it and gave me very useful and interesting comments.

The final thanks go to all my old friends (from childhood, high school and university), with whom I continue to have a very nice relationship, even though we rarely see each other. The fact that you were mentioned in my two former theses, and that you also found a place in the third one should count for something.

I leave the reader with a quote which is really inspiring to me

"Choose a job you love, and you will never have to work a day in your life."
Old wiseman