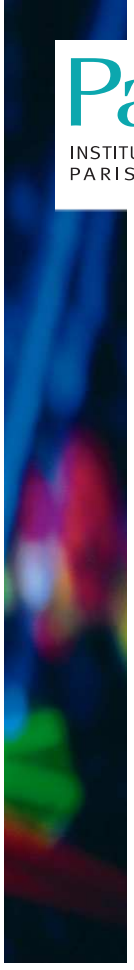


ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0078

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Electronique et Communications »

présentée et soutenue publiquement par

Franck IUTZELER

le Vendredi 6 Décembre 2013

Estimation et Optimisation Distribuée pour les réseaux asynchrones

Directeurs de thèse : **Philippe CIBLAT** et **Walid HACHEM**

Jury

M. Cédric RICHARD, Professeur, Université de Nice Sophia-Antipolis	Président du jury
M. Michael RABBAT, Associate Professor, McGill University	Rapporteur
M. Julien HENDRICKX, Chargé de Cours, Université catholique de Louvain	Rapporteur
M. Geert LEUS, Professor, Delft University of Technology	Examinateur
M. Pierre BORGNAT, Chargé de Recherche, Ecole Normale Supérieure de Lyon	Examinateur
M. Philippe CIBLAT, Professeur, Telecom ParisTech	Directeur de Thèse
M. Walid HACHEM, Directeur de Recherche, Telecom ParisTech	Directeur de Thèse
M. Pascal BIANCHI, Maître de Conférences, Telecom ParisTech	Invité

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Distributed Estimation and Optimization in Asynchronous Networks

Franck IUTZELER

REMERCIEMENTS

Je voudrais tout d'abord remercier Philippe Ciblat et Walid Hachem sans rien de tout cela ne serait arrivé. Leur rigueur et leur enthousiasme pour la recherche ont été très importants pour moi. Un grand merci également à Pascal Bianchi et Jérémie Jakubowicz pour ces échanges et collaborations fructueuses.

Je tiens particulièrement à remercier Julien Hendrickx et Michael Rabbat pour avoir accepté de rapporter ma thèse ainsi que pour leurs commentaires toujours pertinents. Je remercie également Cédric Richard, qui a été le président de mon jury ainsi que Geert Leus et Pierre Borgnat, examinateurs lors de ma soutenance.

Je suis aussi très reconnaissant envers :

- les doctorants et post-doctorants du département COMELEC dont l'énumération serait aussi fastidieuse qu'inutile. Parmi ces derniers, merci particulièrement à Germain, Chadi, Seb, Pierre, et enfin Mélanie qui a eu une place bien particulière dans ma thèse et ma vie.
- les amis "hors-thèse" (même si beaucoup sont ou seront docteurs) Florian, Moyen Sam, Jérôme, Kader, Jean-Jacques, Béné, Léopold, Popol et Marguerite.
- les HHQQP et plus particulièrement Flo, Axelle, Victor, Jon, Tômo, Alexis et Sam, qui ont réussi a rendre simultanément mes matins plus difficiles et ma vie plus sympa.
- mes amis de Besac Lucie, Laure, Arnold, Romain, Alex, Renaud et Camille.
- mes deux colocs Charlix et Toubi pour ces trois merveilleuses années ensemble.

Finalement, je tiens à remercier ma famille et plus particulièrement ma tante Françoise, ma grand-mère Suzanne, et bien évidemment mes parents pour m'avoir toujours poussé et supporté dans mes projets.

CONTENTS

List of Acronyms

i.i.d.	independent and identically distributed
WSN	Wireless Sensor Network
RGG	Random Geometric Graph
SE	Squared Error
MSE	Mean Squared Error
LRT	Likelihood Ratio Test
LLR	Log-Likelihood Ratio
SNR	Signal-to-Noise Ratio
cdf	cumulative distribution function
ROC	Receiver Operating Characteristic

Algorithms:

BWGossip	Broadcast Weighted Gossip
lasso	least absolute shrinkage and selection operator
ADMM	Alternating Direction Method of Multipliers

General rules for notation

x	vector (generally in $\mathbb{R}^n$ and potentially scalar)
x_i	i-th coefficient of vector x
x^k	vector x at time k
$\mathbb{1}$, $\mathbb{1}_N$	the vector of ones, " of size N
$\mathbf{X}$	matrix
$\mathbf{X} \geq 0$, $\mathbf{X} > 0$	elementwise nonnegative matrix , elementwise positive matrix
$\mathbf{X} \succcurlyeq 0$, $\mathbf{X} \succ 0$	positive semidefinite matrix , positive definite matrix
X	operator
$\mathcal{G} = (V, E)$	a graph seen as vertex set V and an edge set E
V	a vertices/agents set
E	an edge/link set
(i, j)	a directed edge/link from i to j
$\{i, j\}$	an undirected edge/link between i and j

INTRODUCTION

The work presented in this thesis was produced within the Digital Communications group, COMmunications and ELECtronics (COMELEC) department of TELECOM ParisTech between October 2010 and October 2013. It was partially funded by the French Defense Agency (DGA) and Télécom/Eurécom Carnot Institute.

Problem statement

During the past decade, the distributed estimation/computation/detection/optimization issue has received a lot of attention. The reasons are manifold since such topics occur in many different applications from cloud computing to wireless sensor networks via machine learning.

Let us focus on two examples in order to be convinced by the importance of this topic. The first example deals with big data. Assume that we have to treat a large amount of data. No device can handle data this big so the processing has to be split in many subtasks for which the computational load is reasonable. In order to find the global result, the subtasks have to exchange partial outputs and so the decentralized devices (called agents/nodes/sensors, in the following) have to communicate their own results often. In order to avoid network congestion and overhead due to routing algorithm, a fusion center approach is prohibitive and the decentralized devices have to talk locally and in a distributive way. In addition, as each device has its own computational ability (due to the processor, the load, etc.), the local outputs are not available synchronously. So, we need to develop asynchronous distributed algorithms. The second example deals with Wireless Sensor Networks which are mainly consider to sensing and inferring in hostile large-scale environments (mountains, deserts, etc.). Assume that we would like to monitor the temperature (max or mean) of an area. Once again in order to avoid network congestion and overhead due to routing algorithm, no fusion center is available. So the value of interest has to be available at each sensor in a distributive way. Once again, an asynchronous communication scheme is considered since it avoids the use of complicated communications coordination (in time and/or frequency).

Therefore, in this thesis, **we will focus on the computation/optimization of a global value of interest using only local and asynchronous (sometimes wireless) communications**, thus when no fusion center is available. Obviously, the methods dealing with such a problem greatly depend on the value of interest of the network which ranges from a particular

sensor measurement (the problem then simply consists in spreading this value) to the output of a complex optimization problem. Therefore we address in this thesis three very different problems.

- **Problem 1:** how to find and spread the maximal initial value of the sensors throughout the network; this problem will enable us to understand how a piece of information spreads over in a network.
- **Problem 2:** how to compute the average of the initial values of the sensors; this problem will enable us to understand how to merge local information in a linear way.
- **Problem 3:** how to optimize a global function when each sensor can only compute optimization of its own local cost function. To do so, we assume that the global function is the sum of the local cost functions.

Outline and Contributions

Before focusing on the above-mentioned problems, in Chapter 1, we introduce, on the one hand, the considered model for Wireless Sensor Networks where the network of agents and their link are seen as a graph and, on the other hand, we provide some essential properties about non-negative matrices and their relations with graphs. Notice that one of the main important assumption done throughout the thesis is the asynchronism between the nodes communications.

In Chapter 2, we will study the problem of finding the maximal value in a wireless network. More precisely, take a sensor network where every agent owns a scalar measurement, we want the network to exchange through their wireless links (which ultimately means that they can broadcast) in order to make the maximum value known everywhere in the network. This problem seems artificial but it is very useful to understand the propagation of information across the network. Actually, our problem is quite close to the well-known rumor spreading problem in an asynchronous radio network; however, a major difference exists as in the rumor spreading problem the nodes of the network know if they have the wanted value or not, which is not the case for us. We will analyze this problem through three very different algorithms: a simple random walk, a random algorithm based on pairwise communications, and a random algorithm based on broadcasting. Whereas the random walk which has been studied extensively in the literature, the two other algorithms remain untreated. Our contribution to this problem is thus to analyze the performance of these two algorithms by deriving bounds on their mean convergence time and their convergence time dispersion. These new results enable us to compare with similar algorithms from the rumor spreading literature and give considerations about the performance gain due to the use of broadcast communications.

In Chapter 3, we will focus on distributed averaging algorithms. Given initial nodes measurements, these algorithms aim at making every sensor learn the average of the initial measurements in a distributive way. So each sensor performs local, cleverly weighted averages and exchanges the results of these averages with their neighbors. This topic has received a

lot of interest in the past ten years; one of the first and most famous algorithms is called random gossip and simply consists in selecting a random pair of neighbors which will average their values. Broadcast-based algorithms have also been developed but they suffer either from collision-related issues due the need of feedback, or from convergence issues. Thanks to the results reminded in Chapter 1, we will review the convergence (or non-convergence) results of the literature. As we exhibited the benefits of broadcast communications on information spreading in Chapter 2, we will then focus on the class of averaging algorithms where feedback links are not mandatory[1] (broadcast communications are a special case). Recently, Bénézit *et al.* proved that by updating identically two variables per sensor, one initialized to the sensor measurement and the other to 1, one could design feedback-free algorithm such that the ratio of these two values converged to the wanted average for every sensor. Actually, this setup based on two variables is called *Sum-Weight* and was introduced by Kempe in 2003. The main contributions of this Chapter are twofold: i) we design a new broadcast-based averaging algorithm called Broadcast Weight Gossip (BWGossip) which outperforms existing algorithms. And, ii) we prove that algorithms based on Sum-Weight set-up converge with an exponential rate (and a bound on this rate is also characterized) under mild assumptions. Notice that our results apply to existing gossip algorithms (with feedback and a single variable per sensor). We show by numerical simulations that the obtained bound is very tight for both our algorithm and existing ones. Finally, we remark that the Sum-Weight scheme enables us to distributively compute the sum without any information about the number of nodes in the network. As an application example, we consider the problem of signal detection in a cognitive radio network; the secondary users have great interest in collaborating as some may have much more information about the primary transmitters than others. We show that this distributed detection problem ultimately relies on the computation of the average and the sum of the test result of the secondary users. Thanks to our previous considerations we are thus able to design a new fully distributed signal detection algorithm.

In Chapter 4, we will consider the problem of solving a distributed optimization problem, that is a problem where the network wants to know the minimum of the sum of its agents cost functions. This situation often arises while performing data processing over big data network; indeed, the quantity of information stored in physically distant servers has grown to become way too large to be transmitted to a fusion node. Hence, in order to process this whole information, the agents of the network have to cleverly process their own information and exchange some information (a few scalars at most) asynchronously with conveniently close nodes (in order not to flood the network with information). Let f be the convex function, depending on all the data of the network, that we want to know a minimum; we will assume that f can be written as $\sum_i f_i$ where f_i is a convex function depending only on the data of the i-th node. The considered problem is to minimize f over a network where each node i only knows its function f_i. We will first review first order algorithms where each node performs a gradient

[1] We still allow some control back propagation enabling the sender to know if the message was succesfully transmitted to the recipients.

descent on its cost function and then the network performs an average gossip step (as studied in the previous chapter). These algorithms use low computational power but they are quite slow and only use a portion of the information of their cost function (the gradient at a given point); with the increase in the computational abilities, it is pertinent to design algorithms that use a greater part of the sensors information at the expense of a increased computational cost. Recently, the Alternation Direction Method of Multipliers (ADMM) has been proved to perform very well for distributed optimization over networks but in a synchronous setup. Our main contributions here are twofold: i) we develop a new ADMM-based distributed optimization algorithm in the asynchronous setup. And, ii) we obtain the convergence of this algorithm by using the monotone operators framework.

Publications

Journal Papers

J2 **F. Iutzeler**, P. Ciblat, and W. Hachem, "Analysis of Sum-Weight-like algorithms for averaging in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2802–2814, June 2013.

J1 **F. Iutzeler**, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 6103–6107, November 2012.

International Conferences

C4 **F. Iutzeler**, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers," in *IEEE Conference on Decision and Control (CDC)*, December 2013.

C3 **F. Iutzeler** and P. Ciblat, "Fully-distributed spectrum sensing: application to cognitive radio," in *European Signal Processing Conference (EUSIPCO)*, September 2013.

C2 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "New broadcast based distributed averaging algorithm over wireless sensor networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012.

C1 **F. Iutzeler**, J. Jakubowicz, W. Hachem, and P. Ciblat, "Distributed estimation of the maximum value over a Wireless Sensor Network," in *Asilomar Conference on Signals, Systems and Computers*, November 2011.

French Conferences

N2 **F. Iutzeler** and P. Ciblat, "Sondage de spectre coopératif totalement distribué: application à la radio cognitive," in *Colloque GRETSI* , September 2013.

N1 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "Estimation distribuée du maximum dans un réseau de capteurs," in *Colloque GRETSI*, September 2011.

CHAPTER 1

GENERAL MODEL AND TECHNICAL PRELIMINARIES

In this chapter, we will present the mathematical framework used throughout the thesis (e.g., graph theory, nonnegative matrices) as well as some very useful mathematical preliminaries.

1.1 Network Model

1.1.1 Graph-based network

Consider a network of N sensors – often called agents – modeled as a graph $\mathcal{G} = (V, E)$ where V is the set of agents – the vertices of the graph, identified by their indexes – and E is the set of links between agents – the edges of the graph represented as an ordered pair (i, j) if there is a link from i to j. We assume that each link is error-free. The set of *neighbors* of agent i – the nodes to which it can send information – is denoted $\mathcal{N}_i = \{j \in V : (i, j) \in E\}$. For any set S, we denote its cardinality by $|S|$. Obviously, we have $|V| = N$.

When a graph only has bidirectional edges (mathematically, $(i, j) \in E \Rightarrow (j, i) \in E$), it is called *undirected* and we will note the edges $\{i, j\} = (i, j) \cup (j, i)$, whereas in the general case it is *directed*. This has an impact on the reachability of the nodes from one another: in directed graphs, if there is a *path* (*i.e.*a succession of edges) from i to j ($i \neq j$), there may not be one from j to i; in undirected graphs, there is always one. In directed graphs, we will call *weakly connected* a graph such that for any $i, j \in V$ ($i \neq j$), there is either a path from i to j or a path from j to i; and we will call *strongly connected* a graph such that for any $i, j \in V$ ($i \neq j$), there is both a path from i to j and a path from j to i. Obviously, in undirected graphs the two notions are equivalent and we just use the term *connected*. When a node i is connected with itself ($(i, i) \in E$), we say that i has a *self-loop*. Finally, we assign a *weight* $\omega_{(i,j)} > 0$ to any edge $(i, j) \in E$; by default all the weights are set to 1, if it is not the case the graph is said to be *weighted*. One can easily remark that the values of the edges non-null weights do not change anything in the connectivity properties of the graph. In addition, one can define the

adjacency matrix of the graph $\mathbf{A}$ as the matrix such that $\mathbf{A}_{i,j} = \omega_{(i,j)}$ if there is an edge from i to j and zeros elsewhere. Consequently, if the graph is self-loop-free, then each diagonal term of the adjacency matrix is zero. In contrast, if the graph has self-loop, the adjacency matrix has some non-null diagonal terms. Once again, one can easily prove that removing self-loop in a graph does not modify its connectivity property. One can define the *support* of the graph associated with the adjacency matrix $\mathbf{A}$ as the matrix $\mathbf{S} = \text{Supp}(\mathbf{A})$ such that $\mathbf{S}_{i,j} = 1$ if $\mathbf{A}_{i,j} \neq 0$ and zeros elsewhere. Notice that the connectivity of the graph can be checked either on $\mathbf{A}$ or $\mathbf{S}$ equivalently. Moreover, if the graph is unweighted, the adjacency matrix is identical to its support.

Let $\mathscr{G} = (V, E)$ be a N-vertices undirected graph. We also define the *degree matrix* $\mathbf{D}$ as the $N \times N$ diagonal matrix such that $\mathbf{D}_{i,i} = d_i$ with $d_i = \sum_{j \in V} \mathbf{A}_{i,j}$; if the graph is unweighted then $d_i = |\mathscr{N}_i|$ the *degree* of i. Finally, we define the *Laplacian matrix* of the graph $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The eigenvalues of this matrix play a fundamental role in algebraic graph theory, indeed $\mathbf{L}$ is i) positive semidefinite (see for example [1, Section 1.2] or [2]); and ii) its row sum is null so $\mathbb{1}_N$ is an eigenvector associated with eigenvalue 0. Hence, the eigenvalues of $\mathbf{L}$ satisfy $0 = \lambda_1^{\mathbf{L}} \leq \lambda_2^{\mathbf{L}} \leq \ldots \leq \lambda_N^{\mathbf{L}}$ (see [3, Th. 7.2.1]). Furthermore, if $\mathscr{G}$ is connected, then the second smallest eigenvalue of the Laplacian[1] $\lambda_2^{\mathbf{L}}$ is strictly positive (see [1, Lemma 1.7]). One can see that the Laplacian matrix is insensitive to the presence or absence of self-loops in the considered graph. These definitions (adjacency matrix, degree matrix, and Laplacian one) can be extended to the case of directed graph. Since these tools for directed graphs will not be used in the thesis, we omit them (see [5] for more details).

1.1.2 Nodes activations

The network is assumed *asynchronous*, meaning that no common clock is available for the agents. Instead, each agent has its own clock and can only initiate a communication at its clock ticks. Assuming that the communication time is small compared to the time between clock ticks, it makes sense (as usually done for other consensus-like algorithms [6, 7]) to assume the absence of collisions between communicating nodes. In addition, we can consider that the agents clocks are modeled by independent Poisson processes with intensity Λ_i for each agent i. It is then equivalent to having a global Poisson clock with intensity $\Lambda = \sum_i \Lambda_i$ and to attribute each clock tick to an agent. More precisely, the process, indexed by the number of global clock ticks, of the sensors to which the global clock ticks are attributed is thus independent and identically distributed (i.i.d.) and the probability that node i becomes active at time k is equal to $p_i = \Lambda_i / \Lambda$. This newly active sensor will then communicate with some of its neighbors. Given that node i is the activating sensor, the probability that i sends information to j will be denoted by $q_{(i,j)}$.

Let us consider the $N \times N$ matrix $\mathbf{Q}$ whose (i, j)-th entry is equal to $q_{(i,j)}$ which we will call the *communication matrix* of the network. As we want information to spread using the

[1] often called the Fiedler value in dedication to its pioneering work [4], or the spectral gap.

available links, we will make the following assumption throughout the thesis.

Assumption 1.1. *The communication matrix* $\mathbf{Q}$ *is* adapted *to the graph* $\mathcal{G}$*,* i.e. $\mathbf{Q}$ *and* $\mathbf{A}$ *have the same support.*

1.1.3 Nodes measurements

In Wireless Sensor Networks (WSNs), the sensors often have measurements that we will call *initial values* and the goal of the WSN will then be to cooperate in order to compute distributively a function of these initial values (the average, the maximum, etc.). Formally, the initial value of sensor i is denoted by x_i^0 and we define $x^0 = [x_1^0, ..., x_N^0]^{\mathrm{T}}$. After sending or receiving information, a sensor may update its value so we denote by x_i^k the value of sensor i at the k-th global clock tick and $x^k = [x_1^k, ..., x_N^k]^{\mathrm{T}}$.

1.2 Non-negative matrices

In this thesis, the matrices with non-negative entries will play a great role (see matrices $\mathbf{A}$ and $\mathbf{Q}$ defined in the previous sections). Any non-negative matrices can be actually associated with a graph. In the following, we will remind some properties of the non-negative matrices and their associated graphs. *All below-mentioned results can be found in Chapters 6.2 and 8 of [3].*

1.2.1 Definitions

Let $\mathbf{A}$ be a $N \times N$ matrix. We say that $\mathbf{A} \geq 0$ ($\mathbf{A}$ is *non-negative*) if $\forall i, j,\ \mathbf{A}_{i,j} \geq 0$. Similarly, we say that $\mathbf{A} > 0$ ($\mathbf{A}$ is *positive*) if $\forall i, j,\ \mathbf{A}_{i,j} > 0$.

For any $N \times N$ non-negative matrix $\mathbf{A}$, we define its *induced graph* $\mathcal{G}(\mathbf{A})$ as the (directed and weighted) graph of N nodes such that there is a directed edge from i to j if and only if $\mathbf{A}_{i,j} > 0$, its weight of this edge is then $\omega_{(i,j)} = \mathbf{A}_{i,j}$. Consequently, the adjacency matrix of the graph $\mathcal{G}(\mathbf{A})$ is $\mathbf{A}$. So, there is a one-to-one mapping between the set of graphs (and its associated adjacency matrix) and the non-negative matrices.

In Section 1.1.1, we said that the graph is connected if it exists a path from i to j for any i and j. Actually, there exists a path of length $m_{i,j} > 0$ in $\mathcal{G}(\mathbf{A})$ from i to j if and only if $(\mathbf{A}_{i,j}^m)_{i,j} > 0$. Therefore, it is of great interest to inspect the powers of $\mathbf{A}$ and its possible positivity.

1.2.2 Irreducible non-negative matrices

Let $\mathbf{A}$ be a $N \times N$ non-negative matrix. $\mathbf{A}$ is said to be *reducible* if it can be made upper-block-triangular by the same permutations of the columns and rows [3, Definition 6.2.21]. It is said *irreducible* if it is not reducible; furthermore, we know that $\mathbf{A}$ is irreducible if and only if $(\mathbf{I} + \mathbf{A})^{N-1} > 0$. Since $(\mathbf{I} + \mathbf{A})^{N-1} = \sum_{i=0}^{N-1} \binom{N-1}{i} \mathbf{A}^i$, we see that if $\mathbf{A}$ is irreducible, then for any i, j (with $i \neq i$) at least one of the matrices $\mathbf{A}, \mathbf{A}^2, ..., \mathbf{A}^{N-1}$ has a positive (i, j)-th entry. This

means that, for any i, j (with $i \neq i$), there is a path from i to j in $\mathcal{G}(\mathbf{A})$; in other words, $\mathcal{G}(\mathbf{A})$ is strongly connected. Let us summarize:

Proposition 1.2. *Let* $\mathbf{A}$ *be a* $N \times N$ *non-negative matrix. The following are equivalent:*

i) $\mathbf{A}$ *is irreducible;*

ii) $(\mathbf{I}+\mathbf{A})^{N-1} > 0$*;*

iii) $\mathcal{G}(\mathbf{A})$ *is strongly connected.*

Thanks to Proposition 1.2, one can remark that if $\mathbf{A}$ is irreducible, then Supp($\mathbf{A}$) is irreducible too.

A very interesting result about irreducible matrices is *Perron-Frobenius theorem* which states that if $\mathbf{A}$ is *irreducible*, then its *spectral radius* $\rho(\mathbf{A}) \triangleq \max_i\{|\lambda_i^{\mathbf{A}}|\}$ is a simple eigenvalue of $\mathbf{A}$ associated with a positive vector.

Theorem 1.3 (Perron-Frobenius theorem for irreducible non-negative matrices)**.** *Let* $\mathbf{A}$ *be a* $N \times N$ *non-negative matrix and suppose that* $\mathbf{A}$ *is irreducible. Then*

i) $\rho(\mathbf{A}) > 0$ *is a simple eigenvalue of* $\mathbf{A}$*;*

ii) there is a positive vector x *such that* $\mathbf{A}x = \rho(\mathbf{A})x$.

1.2.3 Primitive non-negative matrices

Let $\mathbf{A}$ be a $N \times N$ non-negative matrix. We say that $\mathbf{A}$ is *primitive* if and only if it exists $m \geq 1$ such that $\mathbf{A}^m > 0$. First of all, since $(\mathbf{I}+\mathbf{A})^{N-1} = \sum_{i=0}^{N-1} \binom{N-1}{i} \mathbf{A}^i$ and any $\mathbf{A}^m$ with $m \geq N$ is a linear combination of $\{\mathbf{A}^n\}_{n=0,\ldots,N-1}$ (see Cayley-Hamilton polynomial), we obtain that a primitive matrix is irreducible which implies that $\mathcal{G}(\mathbf{A})$ is strongly connected. Consequently the primitivity property is stronger than the irreductibility one. So there are some fundamental differences between irreducible and primitive matrices and their associated graphs. The main difference is as follows: one can prove that, if $\mathbf{A}$ is primitive (related to the power m), then $\mathbf{A}^k > 0$ for any $k \geq m$. Consequently, for any $k \geq m$, and any i, j, there is a path from i to j of length k in $\mathcal{G}(\mathbf{A})$. It is not the case for graphs induced by irreducible matrices. Notice also that a self-loop-free graph associated with an irreducible matrix can induce a primitive matrix very easily since its self-loop related graph (a self-loop is added for each node) has a primitive adjacency matrix. The next proposition will be very useful in the remainder of the thesis.

Proposition 1.4. *Let* $\mathbf{A}$ *be a* $N \times N$ *non-negative matrix. If* $\mathbf{A}$ *is irreducible and has positive diagonal entries, then* $\mathbf{A}$ *is primitive.*

Once again, one can see that if $\mathbf{A}$ is primitive, then Supp($\mathbf{A}$) is primitive too.

Perron-Frobenius theorem can be slightly improved for primitive matrices. Indeed, a useful property can be added compared to the case of irreducible matrices: the eigenvalue $\rho(\mathbf{A})$ is the only eigenvalue of maximal modulus.

Theorem 1.5 (Perron-Frobenius theorem for primitive matrices)**.** *Let* $\mathbf{A}$ *be a* $N \times N$ *non-negative matrix and suppose that* $\mathbf{A}$ *is primitive. Then*

i) $\rho(\mathbf{A}) > 0$ *is a simple eigenvalue of* $\mathbf{A}$*;*
ii) there is a positive vector x *such that* $\mathbf{A}x = \rho(\mathbf{A})x$*;*
iii) $\rho(\mathbf{A})$ *is the only eigenvalue of maximal modulus.*

Actually, a stronger result holds: if $\mathbf{A}$ is irreducible but not primitive, then there exists an integer k_A strictly greater than one, such that the matrix $\mathbf{A}$ has k_A eigenvalues of maximal modulus [3, Chap 8.4 and 8.5]. *This property will be a major role in the following and explains why the primitivity property will be often required.*

1.2.4 Stochastic matrices

A non-negative $N \times N$ matrix is said to be *row-stochastic* if the sum of each row is equal to one, *i.e.*,

$$\mathbf{A}\mathbb{1} = \mathbb{1} \quad \text{(row-stochastic)}.$$

A non-negative $N \times N$ matrix is said to be *column-stochastic* if the sum of each column is equal to one, *i.e.*,

$$\mathbb{1}^{\mathrm{T}}\mathbf{A} = \mathbb{1}^{\mathrm{T}} \quad \text{(column-stochastic)}.$$

A non-negative $N \times N$ matrix is said to be *doubly-stochastic* if it is row-stochastic as well as column-stochastic.

In the following, we remind very useful results about such matrices. First of all, the spectral radius of a row-stochastic or column stochastic matrix is equal to one which means that any eigenvalue has a magnitude (non-strictly) smaller that one. The proof is given in [3, Lemma 8.1.21]. This property will also play a major role in our derivations.

Theorem 1.6. *Let* $\mathbf{A}$ *be a either row-stochastic or column-stochastic matrix. Then, we have*

$$\rho(\mathbf{A}) = 1.$$

Consequently, the vector $\mathbb{1}$ is a (left or right)-eigenvector associated with the largest eigenvalue. In the remainder of the thesis, the orthogonal projection on the span of $\mathbb{1}$ will be often used. Let $\mathbf{J}$ be this projection matrix. Then we have

$$\mathbf{J} = \frac{1}{N}\mathbb{1}\mathbb{1}^{\mathrm{T}}.$$

In addition, let $\mathbf{J}^{\perp}$ be the projection matrix on the orthogonal of span($\mathbb{1}$). We easily have that

$$\mathbf{J}^{\perp} = \mathbf{I} - \mathbf{J}.$$

Proposition 1.7. *Let* $\mathbf{A}$ *be a row-stochastic matrix. We have*

$$\mathbf{J}^{\perp}\mathbf{A}\mathbf{J}^{\perp} = \mathbf{J}^{\perp}\mathbf{A}.$$

Proposition 1.8. *Let* $\mathbf{A}$ *be a column-stochastic matrix. We have*

$$\mathbf{J}\mathbf{A} = \mathbf{J} \quad \textit{and} \quad \mathbf{J}^{\perp}\mathbf{A} = \mathbf{A} - \mathbf{J}.$$

1.2.5 Stochastic matrices and Markov Chain

First of all, any $N \times N$ row-stochastic matrix can be viewed as a transition probability matrix of a discrete-time Markov Chain with N states and conversely.

Let $\mathbf{W}_k$ be the transition probability matrix of a discrete-time non-homogeneous Markov Chain at time k. If t^k is the distribution over the states at time k, then, at time $k+1$, the new distribution over the states is denoted by t^{k+1} and is given by

$$t^{k+1^{\mathrm{T}}} = t^{k^{\mathrm{T}}} \mathbf{W}_k \tag{1.1}$$

Since t^k corresponds to a distribution, t^k is a non-negative column-vector such that $\mathbb{1}^{\mathrm{T}} t^k = 1$.

Obviously, for analyzing the evolution in time of the states distribution, the aggregate of the row-stochastic matrices $\mathbf{W}_k$ has to be done as follows

$$\mathbf{P}_f^{1,k} = \mathbf{W}_1 \mathbf{W}_2 \cdots \mathbf{W}_k.$$

We thus have to operate right matrix multiplications, and this direction is called *forward*. Obviously, the analysis of the matrix $\mathbf{P}_f^{1,k}$ is of great interest in order to inspect the asymptotic behavior of the associated Markov Chain.

1.2.5-a Forward direction

To handle the analysis of the forward matrix $\mathbf{P}_f^{1,k}$, the notion of (weak or strong) ergodicity has to be introduced.

Definition 1.9. *A sequence of $N \times N$ row-stochastic matrices $\{\mathbf{W}_k\}_{k>0}$ is said to be* **weakly ergodic** *if for all $i, j, \ell \in \{1, ..., N\}$ and all $s > 0$*

$$\left[\left(\mathbf{P}_f^{s,s+k} \right)_{i,\ell} - \left(\mathbf{P}_f^{s,s+k} \right)_{j,\ell} \right] \overset{k \to \infty}{\longrightarrow} 0.$$

where

$$\mathbf{P}_f^{s,s+k} = \mathbf{W}_s \mathbf{W}_{s+1} \cdots \mathbf{W}_{s+k}.$$

In other words, weak ergodicity means that the rows of the aggregative matrix tend to be identical, **but** may vary with k. By abuse of notations, we write that weak ergodicity implies that it exists a sequence of non-negative column vector v^k (with $\mathbb{1}^{\mathrm{T}} v^k = 1$) such that

$$\mathbf{P}_f^{1,k} \overset{k \to \infty}{\sim} \mathbb{1} v^{k^{\mathrm{T}}} \tag{1.2}$$

where the notation $\overset{k \to \infty}{\sim}$ stands for two terms are equivalent for k large enough. As conclusion, when we will apply such a result to our consensus algorithms, we will obtain that the nodes agree but do not reach a consensus since the agreement value changes at each iteration.

A stronger definition of the ergodicity exists in the literature. If this type of ergodicity holds, then the nodes reach a consensus.

Definition 1.10. *A sequence of $N \times N$ row-stochastic matrices $\{\mathbf{W}_k\}_{k>0}$ is said to be* **strongly ergodic** *if there is a non-negative column vector v (with $\mathbb{1}^{\mathrm{T}} v = 1$) such that for all $s > 0$*

$$\mathbf{P}_f^{s,s+k} \overset{k\to\infty}{\longrightarrow} \mathbb{1} v^{\mathrm{T}}. \tag{1.3}$$

In other words, strong ergodicity means that the rows of the aggregative matrix tend to the same vector v^{T} which represents actually a distribution. Clearly strong ergodicity implies weak ergodicity (see [8, Chap. 6.8]).

Remark 1.11. *If the considered matrices $\{\mathbf{W}_k\}_{k>0}$ are column-stochastic (and not row-stochastic), then all the above-mentioned results still hold by replacing $\mathbf{W}_k$ with $\mathbf{W}_k^{\mathrm{T}}$ since $\mathbf{W}_k^{\mathrm{T}}$ is a row-stochastic matrix.*

Remark 1.12. *If the considered matrices $\{\mathbf{W}_k\}_{k>0}$ are doubly-stochastic, then all the above-mentioned results still hold. In addition, we have $v = (1/N)\mathbb{1}$ in Definition 1.10.*

1.2.5-b Backward direction

As we will see later in the thesis (in Chapter 3), we also have to work with the so-called backward direction defined as follows

$$\mathbf{P}_b^{1,k} = \mathbf{W}_k \mathbf{W}_{k-1} \cdots \mathbf{W}_1$$

where $\{\mathbf{W}_k\}_{k>0}$ are a set of $N \times N$ row-stochastic matrices.

Obviously, the weak ergodicity and the strong one can defined similarly to previous Subsection by replacing $\mathbf{P}_f^{s,s+k}$ with $\mathbf{P}_b^{s,s+k}$. In that case, it has been surprisingly proven that the weak ergodicity is equivalent to the strong one [9].

1.2.5-c Particular case: homogeneous Markov Chain

A Markov Chain is said homogeneous when the transition probability matrix is independent of time k. So, we then can work with an unique row-stochastic matrix, denoted here by $\mathbf{W}$. In that case, we have $\mathbf{P}_f^{s,s+k} = \mathbf{P}_b^{s,s+k} = \mathbf{W}^{k+1}$ and the forward direction is equivalent to the backward one. In addition, the weak ergodicity is equivalent to the strong one by applying the result given in Subsection 1.2.5-b. Therefore, in homogeneous case, the adjectives *weak*, *strong*, *forward*, and *backward* are omitted.

In homogeneous case, if a non-negative column-vector π (with $\mathbb{1}^{\mathrm{T}}\pi = 1$) is a left-eigenvector of $\mathbf{W}$ associated with the eigenvalue equal to 1, *i.e.*,

$$\pi^{\mathrm{T}}\mathbf{W} = \pi^{\mathrm{T}},$$

then the vector represents a stable distribution of the associated Markov Chain.

In addition if the Markov Chain is ergodic, then

$$\pi = v \tag{1.4}$$

where v is defined as in Definition 1.10.

1.3 Norms

Here, we just recall the definitions of two norms associated with matrices (see [3, Chapter 5] for details).

Let $x = [x_1, \cdots, x_N]^{\mathrm{T}}$ be a $N \times 1$ vector. The Euclidian (or l_2) norm is defined as

$$\|x\|_2 = \sqrt{\sum_{n=1}^{N} |x_n|^2}.$$

Let $\mathbf{A}$ be any $N \times N$ matrix. The spectral (or l_2-induced) norm of this matrix, denoted by $|||\cdot|||_2$, is defined as follows

$$|||\mathbf{A}|||_2 = \max_{x \in \mathbb{C}^N} \frac{\|\mathbf{A}x\|_2}{\|x\|_2} = \max\left\{\sqrt{\lambda} : \lambda \text{ is an eigenvalue of } \mathbf{A}^{\mathrm{T}}\mathbf{A}\right\}.$$

The Frobenius norm, denoted by $\|\cdot\|_F$, is defined as

$$\|\mathbf{A}\|_F = \sqrt{\mathrm{Trace}\,(\mathbf{A}^{\mathrm{T}}\mathbf{A})} = \sqrt{\sum_{i,j=1}^{N} |\mathbf{A}_{i,j}|^2}.$$

CHAPTER 2

MAXIMAL VALUE SPREADING

In this chapter, we analyze different techniques for distributively spreading the greatest of the initial values in the context of an asynchronous Wireless Sensor Network. This quite simple question can arise in many applications and it also sheds light on the speed of information dissemination in an asynchronous network.

2.1 Introduction

Wireless Sensor Networks can often be seen as groups of connected agents that cooperate to achieve a common goal. As they are generally deployed in hostile environments, there can not be a master agent acting as a fusion center fetching the data and coordinating the agents; hence, they have to exchange locally with each other in order to achieve their objective.

For example, if a network has to transmit periodically some data through a costly link, a natural way to operate would be to elect the sensor with the largest amount of power remaining to operate that communication. To do so, the sensor network has to retrieve and spread the maximal amount of energy left in the sensors battery (along with the ID of the corresponding sensor); this has to be done distributively using only the wireless links between some of them.

Another useful application deals with distributed medium access control: in the case where many nodes want to send information using the same medium, the network has to choose which node will effectively transmit. A possible distributed protocol is to make the agents that want to send i) draw a number in a common window; and then ii) reach consensus over the greatest drawn value (and the ID of the associated agent). The sensor with the maximal value then sends its packet to the access point.

We remark that all these actions require the computation of the maximum between the initial values of the network. Mathematically, we need an asynchronous distributed algorithm which produces a sequence $\{x^k\}$ such that

$$x^k \longrightarrow x_{\max}\mathbb{1} \tag{2.1}$$

with $x_{\max} \triangleq \max_i x_i^0$. Obviously, all maximal value spreading algorithms easily generalize to the computation of the minimum value, the closer to a constant, etc.

Interestingly, if the sensor with the maximal value knows that it has the maximum across the network, then our problem reduces to the well-known *rumor spreading* problem where the rumor bearing sensor spreads it to the others using the underlying communication network. Our setup, where the agents of the network do not know if they have the rumor to spread or not, can be seen as a *juiciest rumor spreading* in which the network has to locate the juiciest rumor before spreading it. Another fundamental difference with standard rumor spreading is that most rumor spreading algorithms are designed in the context of a synchronous wired networks whereas our setup is an asynchronous wireless network. However, it will be informative to compare the performance of asynchronous maximum value spreading algorithms on wireless networks with similar rumor spreading procedures.

This chapter is organized as follows. Section 2.2 will present the rumor spreading problem and its similitudes and differences with our problem; it will also introduce the algorithms that we will consider for analysis. The details about the setup and how to investigate the algorithms convergence will be given in Section 2.3. Then, Section 2.4 will be dedicated to the mathematical analysis of the mentioned algorithms. Finally, numerical results will illustrate our claims in Section 2.5 and conclusions will be drawn in Section 2.6.

2.2 The maximum value diffusion seen as a Rumor Spreading problem

2.2.1 The rumor spreading problem

Rumor spreading is the problem of spreading a particular, local information through a network. It was introduced by [10] in the case of replicated databases (name servers, yellow pages, etc.) where the updates are made locally (on a single database) and have to be propagated to all other sites. This seminal paper introduced some very useful concepts:

- *rumor mongering* – it is the idea that all updates are not of equal importance to the network so that only the most recent ones may be propagated. It is clear that this asymmetry is something we encounter when trying to spread the maximal initial value through a network.
- *push transmission* – it is when information goes from the caller to the called agent. This transmission setup is responsible for the term *epidemic algorithms* which is the original term coined by Demers *et al.*[10].
- *pull transmission* – it is when information goes from the called to the calling agent.

The push and pull transmission methods are often combined to form *push&pull algorithms*. These algorithms are based on pairwise exchanges between neighbors in the network which recalls the *Random Gossip* algorithm [11] introduced for average value computation and in a asynchronous context.

Due to i) the wired medium in the original works which implies that an agent may only call (or be called by) its neighbors one by one; and ii) the possible congestion and collisions in the network, the rumor spreading algorithms are randomized by nature.

2.2.2 Rumor spreading algorithms

In terms of communication between agents, most papers dealing with rumor spreading consider push and/or pull transmission methods [12, 13, 14, 15, 16, 17, 18, 19] as it is more suited for a wired network. Some papers [20, 21, 22, 23, 24] consider a *radio* medium and hence propose *broadcast*[1] based algorithms.

Randomized rumor spreading may seem very close to our problem but the two problems differ on two major points summarized in Table 2.1. On the one hand, even if the communications of rumor spreading are randomized, many nodes may communicate simultaneously which leads to collisions. In the WSN setup, communications are asynchronous so collisions are avoided by construction and only one sensor can speak at a time. On the other hand, in the rumor spreading problem, the nodes know if they have the desired rumor and hence can act consequently. In the maximal value problem, no sensor can know if it has the maximum value so the time spent by each (potentially useless) communication has to be taken into account. These two differences make our problem quite different from classical rumor spreading and imply the use of different mathematical tools for analyzing the related algorithms. The fact that the sensors do not know if they have the maximal value in our context is a key difference with rumor spreading as it will change the algorithms as well as the proof techniques.

	Rumor spreading	**Maximum value diffusion**
Communication timing	Synchronous. Many nodes may speak at each clock tick. Collisions.	Asynchronous. One node speaks per clock tick. No collisions
Rumor awareness	Aware.	Unaware.

Table 2.1: Key differences between rumor spreading and maximal value spreading in WSNs.

The maximal value diffusion can thus be seen as an "asynchronous juiciest rumor spreading problem" where i) *asynchronous* is linked to the communication framework of WSNs; and ii) *juiciest* means the agents do not know if they own the rumor. Consequently, we need to develop gossip algorithms (*i.e.* algorithms suited for WSNs) solving this problem.

2.2.3 Maximum value gossipping algorithms

To the best of our knowledge, in the framework of distributed computation for WSNs, only [25] has focused on the maximal value computation. Actually, [25] developed a general framework

[1] the term *broadcast* means here sending information to all its neighbors and not spreading information across a network.

to compute a wide family of functions (including the maximum value) of the nodes initial values in a distributed fashion. Compared to our setup, this work has been done under continuous time and synchronous clocks assumptions. It can nevertheless be adapted to our context (discrete time and asynchronous clocks), but the derived algorithm will perform poorly since each node goes to the maximum in an incremental way even if one of its neighbors has the maximum value. Therefore, we proposed gossip algorithms adapted to maximal value computation in Wireless Sensor Networks.

2.2.3-a Random-Walk

First, as a toy example, we consider a simple random walk on the graph propagating the maximal encountered value. This algorithm is not really asynchronous nor synchronous and will be a comparison point as it is very popular and well studied [26, 27].

Random-Walk

At time k, let i be the active node:

- i sends x_i^k to a neighbor j uniformly chosen in $\mathcal{N}_i$;
- j updates: $x_j^{k+1} = \max\left(x_i^k, x_j^k\right)$;
- j is then the active node for time $k+1$.

2.2.3-b Random-Pairwise-Max

A simple way to estimate the maximal value would be to mimic the Random Gossip algorithm introduced for averaging [11]. The agents would wake up randomly and exchange their value with another reachable sensor chosen uniformly; both sensors would then keep the maximum between their former and received values. This can also be seen as an asynchronous *push&pull* algorithm.

Random-Pairwise-Max

At each clock tick k, let i be the activating node:

- i chooses a neighbor j uniformly in $\mathcal{N}_i$ and they exchange their values.
- Both i and j update: $x_i^{k+1} = x_j^{k+1} = \max\left(x_i^k, x_j^k\right)$.

This algorithm is suited for wired networks whereas it is clearly not optimal for wireless networks. Indeed, it does not take advantage of the broadcasting abilities of the wireless channel.

2.2.3-c Random-Broadcast-Max

Since the communications between the sensors are wireless in our context, it seems more natural for the active sensor to broadcast its value, and then the sensors which have received

the information would update their value accordingly. Note that an averaging algorithm based on broadcast communications has been proposed in [7], but it does not perform well due to the non-conservation of the initial sum. This is not an issue for estimating the maximum value since the maximum value is preserved. The RANDOM-BROADCAST-MAX will be our flagship algorithm.

Random-Broadcast-Max

At each clock tick k, let i be the activating node:

- i broadcasts x_i^k to all its neighbors.
- All the neighbors update: $x_j^{k+1} = \max\left(x_j^k, x_i^k\right)$ for all $j \in \mathcal{N}_i$.

One can remark that if the activating sensor has not received information since the last time it broadcasts, it is useless that it broadcasts as its neighbors are already informed with its current value, it could just stay idle. This minor change does not affect the convergence time (as it is always driven by the independent Poisson clocks of the sensors) but reduces the number of transmission and thus the consumed power. In the following we will not consider this change for the sake of simplicity.

2.3 Setup

2.3.1 Precisions about the setup

We will use the model defined in Section 1.1, that is a WSN of N agents modeled as an undirected connected graph $\mathcal{G} = (V, E)$. For the sake of clarity and without loss of generality, we will consider that the agents activate though an i.i.d. process and at any (global) clock tick the activating agent is chosen uniformly in V.

The underlying graph model is very important as the convergence speed is closely linked to the graph structure and the rumor spreading community focuses on finding relations between the performance of the algorithms and the properties of the underlying graph. The following quantities will be useful to characterize the graph (see Section 1.1 for definition and properties):

- the number of vertices N;
- the maximal degree $d_{\max} = \max_{i \in V} |\mathcal{N}_i|$;
- the second smallest eigenvalue of the Laplacian $\lambda_2^{\mathbf{L}}$;
- the diameter $\Delta_{\mathcal{G}} = \max\{l^{\mathcal{G}}(i,j) : (i,j) \in V^2\}$ where $l^{\mathcal{G}}(i,j)$ is the minimum number of edges needed to connect i to j in $\mathcal{G}$.

Even if we want our performance analysis to depend closely on the graph properties, we do not focus on a particular graph unlike [18] which focuses on expanders and [28] which focuses on regular graphs, both in the rumor spreading context.

2.3.2 The convergence time

We define the convergence time τ as the first time when all the nodes share the maximum of the initial values (at this point they should not change their values), *i.e.*,

$$\tau \triangleq \inf\left\{t \in \mathbb{N} : \forall k \geq t \quad x^k = x_{\max}\mathbb{1}\right\}. \tag{2.2}$$

We will now prove a convergence result ($\tau < \infty$ almost surely) for a vast class of random spreading algorithms. Let us consider the class of *Random Maximum Spreading* algorithms, that is the algorithms such that:

- each sensor has one variable initialized with their measurements;
- the activating sensors are chosen through an irreducible Markov process;
- at any time $k > 0$, the newly active sensor sends its variable to some of its neighbors and may also receive the variables of some of its neighbors;
- when a sensor receive a variable greater than its, it updates by replacing its variable by the received one.

Theorem 2.1. *Let Assumption 1.1 hold. Then, for any* Random Maximum Spreading *algorithm, one has $\tau < \infty$ with probability* 1.

Proof. Let $M^k = \{i \in V : x_i^k = x_{\max}\}$ be the set of nodes sharing the maximum at time k. Let $X^k = |M^k|$ be the cardinal of M^k and $Y^k = \delta_{\{X^{k+1} > X^k\}}$ be the random variable equal to 1 when $X^{k+1} > X^k$ and 0 otherwise where $\delta_{\{\cdot\}}$ is the Kronecker symbol.

The considered algorithms are such that M^k is a nondecreasing sequence of subsets of V so X^k is a nondecreasing sequence of integers upper-bounded by N and X^k then converges to a random variable $X^\infty \leq N$. Hence, we have that $\sum_{k=1}^{\infty} Y^k < X^\infty \leq N$ and, taking expectations over the choice of the activating and receiving sensors, $\sum_{k=1}^{\infty} \mathbb{E}[Y^k] < \infty$.

Whenever $X^k < N$, using the graph connectedness, there is at least one couple $(i, j) \in M^k \times (V - M^k)$ such that $j \in \mathcal{N}_i$. For any pair of sensors $i, j \in V$, the probability that i informs j with $x_{\max}$ before N iterations is positive and we will denote by $p > 0$ the lower of these probabilities over any pair of sensors (see Section 1.2.2 for more details). Then, for all $k > 0$

$$\mathbb{E}[\sum_{t=k}^{k+N} Y^t] \geq \mathbb{P}[\sum_{t=k}^{k+N} Y^t = 1, X^k < N] = \mathbb{P}[\sum_{t=k}^{k+N} Y^t = 1 | X^k < N]\mathbb{P}[X^k < N] \geq p\mathbb{P}[X^k < N].$$

Then, by summing over $k > 0$, we have

$$\sum_{k=1}^{\infty} \mathbb{P}[X^k < N] \leq \frac{N}{p} \sum_{k=1}^{\infty} \mathbb{E}[Y_n] < \infty.$$

Thanks to Borel-Cantelli lemma, we know that $\mathbb{P}[X^k < N \text{ infinitely often }] = 0$. Hence, there is a finite time τ such that $X^\tau = N$ almost surely. □

Then, we obviously have the following result.

Corollary 2.2. *The* RANDOM-WALK*,* RANDOM-PAIRWISE-MAX *and* RANDOM-BROADCAST-MAX *converge to a consensus over* $x_{\max}$ *in finite time with probability* 1.

This result is not surprising and we would like now to have more information about the behavior of τ versus some characteristics of the operating graph. For this purpose, we will measure the performance of the proposed algorithms in the light of two criteria:

- the *mean convergence time*: $\mathbb{E}[\tau]$ gives a first order approximation of the convergence speed;
- the *convergence time dispersion*: for all $\varepsilon \in]0,1[$, finding a bound B such that $\mathbb{P}[\tau < B] \geq 1-\varepsilon$ gives a more general view of the convergence speed through the tail of the convergence time distribution.

Finally, in order to be able to compare with the results of the rumor spreading literature, we will study the behavior of our bounds with respect to the number of nodes N when it goes to infinity. To this purpose, we will use the notation $\mathcal{O}(\cdot)$ to get results of the form $\tau = \mathcal{O}(f(N))$ meaning that there is a finite constant C such that $\tau \leq Cf(N)$ for N large enough.

2.4 Performance Analysis

In this section, we will derive performance bounds for the introduced algorithms by the criteria introduced in the previous section.

2.4.1 Random-Walk

This algorithm has mainly a comparison purpose and hence its performance will not be studied extensively. Still, we will relate its performances to some well-studied quantities and give pointers to find results about the expectation and the tail inequalities for these quantities enabling the reader to derive the mean convergence time and convergence time dispersion of the algorithm.

Let us denote by v^k the active node at time k. We define the *walk matrix* $\mathbf{W}$ as $\mathbf{W}_{i,j} = \mathbb{P}[v^{k+1} = j \mid v^k = i]$ for any $i, j \in V$. This matrix has the same support as the communication matrix defined in Section 1.1.2 and thus Assumption 1.1 implies that $\mathbf{W}$ is irreducible by Proposition 1.2. In addition, it is a stochastic matrix *i.e.*a non-negative matrix whose row sum is 1; hence $\mathbf{W}$ can be seen as the transition matrix of N-states irreducible Markov chain. The random walk on a graph can thus be seen as the visit of the states of a stationary Markov chain of transition matrix $\mathbf{W}$.

Let us imagine that agent j has the maximal value at time 0 and that the Random-Walk algorithm begins at sensor $v^0 = i$. Then, the convergence time of the Random-Walk algorithm is the sum of i) the time the walk takes to go from i to j, namely the (i,j)-*hitting time* $H^{\mathbf{W}}(i,j)$; and ii) the time the algorithm takes to go through all the other vertices of the graph starting from j (or identically all the states of the Markov chain), namely the j-*cover time* $C^{\mathbf{W}}(j)$.

Finally, as we want our results to be true for any initialization, we have the following result.

Proposition 2.3. *For the* RANDOM-WALK *algorithm, one has*

$$\tau \leq H^{\mathbf{W}} + C^{\mathbf{W}}$$

where:

- $H^{\mathbf{W}} = \max_{i,j} H^{\mathbf{W}}(i,j)$ *is the maximal hitting time on* $\mathbf{W}$*;*
- $C^{\mathbf{W}} = \max_i C^{\mathbf{W}}(i)$ *is the maximal cover time on* $\mathbf{W}$*.*

From this proposition, the reader can find bounds on $\mathbb{E}[\tau]$ in [26, 29] for general graph and in [27, 30] in the case of Random Geometric Graphs. Concentration bounds on expander graphs can be found in [31].

2.4.2 Random-Pairwise-Max

2.4.2-a Expected convergence time

Theorem 2.4. *For* RANDOM-PAIRWISE-MAX*, one has*

$$\mathbb{E}[\tau] \leq N d_{\max} \frac{h_{N-1}}{\lambda_2^{\mathbf{L}}},$$

with $h_n = \sum_{k=1}^{n} 1/k$ *the n-th harmonic number.*

The details of the proof are reported in Appendix A.2. The idea of the proof is that considering the set S of the nodes informed by the maximum value, the probability for another sensor to become informed after an iteration of Random-Pairwise-Max is the same as the probability that one of the exchanging nodes is in S while the other is in $V \setminus S$. This probability is lower-bounded by $p_{\min}|\partial S|$ where i) $p_{\min} = 1/(N d_{\max})$ is a lower bound of the probability for any pair of connected nodes to be active at each iteration of the algorithm; and ii) $\partial S = \{\{i,j\} \in E : i \in S, j \notin S\}$ is the set of edges with one end in S and the other end in $V \setminus S$. Then, a useful inequality proved in Appendix A.1 is

$$\frac{|\partial S|}{|S|} \geq \lambda_2^{\mathbf{L}} \left(1 - \frac{|S|}{N}\right). \tag{2.3}$$

For the sake of clarity in the comparisons with the literature, it is useful to derive a slightly different version of our bound depending on the *vertex expansion* $\alpha_{\mathscr{G}}$ of the graph instead of $\lambda_2^{\mathbf{L}}$. This quantity is defined as

$$\alpha_{\mathscr{G}} = \min_{S \subseteq V} \frac{|\delta S|}{\min\{|S|, N - |S|\}}$$

where S is any subset of the nodes of $\mathscr{G}$ and $\delta S = \{i \in V \setminus S : \exists j \in S \text{ such that } \{i,j\} \in E\}$ is the set of vertices in $V \setminus S$ with at least one neighbor in S.

Theorem 2.5. *For* RANDOM-PAIRWISE-MAX*, one also has*

$$\mathbb{E}[\tau] \leq N d_{\max} \frac{h_{\lceil \frac{N}{2} \rceil}}{\alpha_{\mathscr{G}}},$$

with $h_n = \sum_{k=1}^{n} 1/k$ *the n-th harmonic number and* $\alpha_{\mathscr{G}}$ *the vertex expansion.*

Proof. The proof is very similar to the one of Theorem 2.4 in Appendix A.2. To obtain the above result, we remark that for any set S, $|\partial S| \geq |\delta S|$, thus Eq. (A.2) still holds with $|\delta S|$ instead of $|\partial S|$. Then, applying the definition of the vertex expansion we have $|\delta S| \geq \alpha_{\mathscr{G}} \min\{|S|, N-|S|\}$ that we plug in the previously mentioned equation to obtain

$$\mathbb{P}[X^{k+1} = X^k + 1 \mid X^k] \geq 2\frac{\alpha_{\mathscr{G}}}{N d_{\max}} \min\{X^k, N - X^k\}$$

and as in Appendix A.2, we sum the inverse of this bound for $X^k = 1, ..., N-1$ to obtain the claimed result. □

In order to illustrate the upper-bound given in Theorem 2.4, let us consider a complete graph[2]; for this particular graph, $d_{\max}/\lambda_2^{\mathbf{L}}$ is of order $\mathscr{O}(1)$ hence our bound is of order $\mathscr{O}(N \log N)$. In the standard rumor spreading context, the bound is of order $\mathscr{O}(\log N)$ [32]. We thus pay an extra factor of order N for not knowing which nodes are informed or not.

2.4.2-b Convergence time dispersion

Theorem 2.6. *For* RANDOM-PAIRWISE-MAX*, with probability* $1-\varepsilon$*,*

$$\tau \leq C^{\mathbb{E}}(N)\left(1 + \log\left(\frac{N}{\varepsilon}\right)\right)$$

where $C^{\mathbb{E}}(N)$ *is one of the bounds (RHS) of Theorem 2.4 or 2.5.*

The proof is reported in Appendix A.3 and is based on concentration inequalities.

Again, in order to compare ourselves with the rumor spreading literature, let us take $\varepsilon = 1/N$, we now have $C^{\mathbb{E}}(N)(1+2\log(N))$ as a right hand side. Hence, $\tau = \mathscr{O}(C^{\mathbb{E}}(N)\log(N))$ with probability $1-1/N$. Using the bound of Theorem 2.5, we get that $\tau = \mathscr{O}(N\alpha_{\mathscr{G}}^{-1}\log^2(N)d_{\max})$ with probability $(1-1/N)$.

In [33], it is proven that τ for the *Push-Pull* is $\mathscr{O}(\alpha_{\mathscr{G}}^{-1}\log^{2.5}(N))$ with probability $(1-1/N)$. Apart from the factor N (essentially due to our communication protocol), the trends offer strong similarities.

2.4.3 Random-Broadcast-Max

2.4.3-a Expected convergence time

Theorem 2.7. *For* RANDOM-BROADCAST-MAX*, one has*

$$\mathbb{E}[\tau] \leq N\Delta_{\mathscr{G}} + N(\Delta_{\mathscr{G}} - 1)\log\left(\frac{N-1}{\Delta_{\mathscr{G}} - 1}\right)$$

where $\Delta_{\mathscr{G}}$ *is the diameter of the graph.*

[2] a *complete* graph is a graph where every vertex is connected to any other one.

The proof is reported in Appendix A.4. In this proof, we actually compute the expected convergence time on a spanning tree subgraph of $\mathcal{G}$ rooted on the node that has the maximum at time 0, that is a graph with the same vertices set but where we delete some edges so that the graph is still connected but has no cycles (or loops, that is sequences of edges starting and ending at the same vertex). Seeing the sensor that has the maximum as the top of the tree, we count the time it needs to inform all its successors (the neighbors, in a descending way) that we call the first *layer* L^1. Once the first layer is informed, we add the time to inform the second layer, etc. When layer L^n is informed, the time to inform L^{n+1} using Random-Broadcast-Max is the time for all the sensors of L^n to become active, this time is equal to $Nh_{|L^n|}$ following the standard proof of the *coupon collector problem* [34].

Note that for complete graphs ($\Delta_{\mathcal{G}} = 1$), the upper bound of Theorem 2.7 is tight since the time needed for propagating the *max* is the time needed for the initially informed node to wake up and communicate its value to all other nodes using only one broadcast communication, hence N in expectation. Moreover, for the ring graph, we can prove that $\mathbb{E}[\tau] = (N^2 - N)/2$ while our bound is close to $N^2(1 + \log(2))/2$, thus both quantities scale in N^2.

Let us now consider the previous works on maximum propagation by using the broadcasting nature of the medium [20, 21, 22, 23, 24]. Even if the framework is strongly different (as showed in Section 2.2), it is interesting to compare the obtained performance bounds. In the rumor spreading framework where all the informed nodes wake up simultaneously and broadcast the information to their neighbors (causing collisions), the expected convergence time behaves like $\Delta_{\mathcal{G}} \log(N/\Delta_{\mathcal{G}})$ [22]. Quite surprisingly, their bound has almost the same shape as ours up to a factor N, which was already observed for the Random-Pairwise-Max (Section 2.4.2-a).

2.4.3-b Convergence time dispersion

Theorem 2.8. *For* RANDOM-BROADCAST-MAX*, with probability* $1 - \varepsilon$*,*

$$\tau \leq C^{\mathbb{E}}(N) + N\Delta_{\mathcal{G}} \left(\log\left(\frac{\Delta_{\mathcal{G}}}{\varepsilon} \right) - 1 \right)$$

where $C^{\mathbb{E}}(N)$ *is the bound of Theorem 2.7.*

The proof is reported in Appendix A.5. It is based on the same framework as the proof of Theorem 2.7 and uses the tail probabilities of the Bernoulli distribution as well as the Union bound.

If we take a look at the complete graph, we remark that the extra time cost scales like $N \log N$ for $\varepsilon = 1/N$. Again, quite surprisingly, [14] obtained similar results although, as before, the two frameworks are strongly different.

2.5 Numerical illustrations

In order to evaluate the performance of the proposed algorithms and the tightness of our bounds, we need to put a model on the underlying graph. Contrary to the rumor spreading literature, we do not want to impose a specific structure on the graph (e.g. to be regular). We thus chose to use Random Geometric Graphs [35] which is a large class of graphs that is very well suited for modeling WSNs as we will see in the following. We will first define and give intuitions about connectivity in RGGs in Section 2.5.1 then we will examine the algorithms performance and the tightness of our bounds in Section 2.5.2.

2.5.1 About RGGs

To obtain such a graph, one has to choose N points uniformly in the unit square $[0,1] \times [0,1]$ (these points represent the position of the nodes/sensors). Then, one has to draw an (undirected) edge between any pair of sensors closer than a pre-defined radius r (this radius represents the communication radius of a sensor, thus the edges link sensors that are able to communicate). The construction of a 10-nodes RGG with radius 0.3 is illustrated in Figure 2.1.

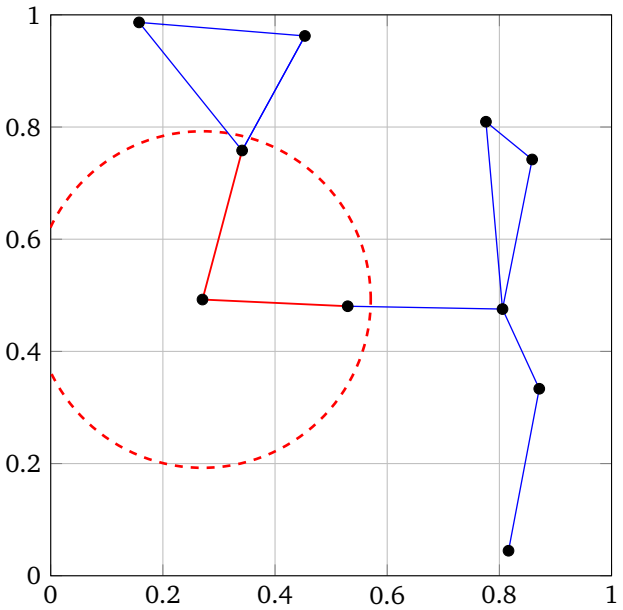
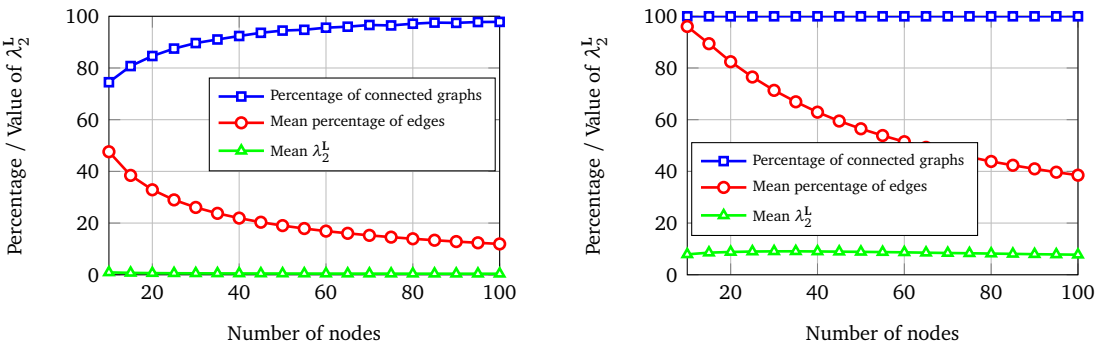


Figure 2.1: Construction of a RGG.

This class of random graphs is well suited for modeling WSNs by their construction and can lead to very different graphs (the circle or the complete graph can be generated with this method with a large variety of intermediaries) which coincide with our objective of finding general bounds valid for any graphs. However, we assumed that our communication graph was connected which may not be the case for some of the RGGs, in this case we just drop the graph (in order to determine if a graph is connected a simple method is to use the properties of Proposition 1.2). By choosing $r = r_0\sqrt{\log(N)/N}$ with r_0 *not too small*, connectivity is ensured

with a high probability (asymptotic connectivity with high probability has been proved for $r_0 \geq 3$, see [36, Appendix A], though smaller radii seem to work quite well for relatively small N).

In Figure 2.2, we plot i) the percentage of connected graphs; ii) the mean percentage of edges (that is the number of edges of the graph over the number of edges of the complete graph with the same number of nodes which is equal to $N(N-1)/2$); and iii) the mean second smallest eigenvalue of the Laplacian $\lambda_2^{\mathbf{L}}$ versus the number of nodes for different values of r_0. We remark that for $r_0 \geq 2$ the graph is always connected for 10 to 100 nodes. We also see that the mean percentage of nodes decreases with N but remains quite high. Finally, the value of $\lambda_2^{\mathbf{L}}$ is very instructive as it is always positive and upper bounded by the number of nodes (bound attained in the case of a complete graph), it also gives an insight on *how well* the graph is connected. This enables us to see how the graphs are more and more connected as r_0 grows, and finally that the graphs generated with $r_0 = 6$ are almost always complete.



(a) r_0=1 (b) r_0=2

(c) r_0=4 (d) r_0=6

Figure 2.2: Connectivity of RGGs with different radii.

2.5.2 About the tightness of the derived bounds

Following what we saw in the previous section, we choose the RGGs with $r_0 = 4$ as a graph model; indeed, they are pretty well connected while at the same time being quite far from the complete graph for $N > 60$ as seen in Fig. 2.2c. All simulations will thus be done over

Monte-Carlo trials of these graphs.

In Figure 2.3, we look at the percentage of informed sensors versus the number of iterations for a 50-sensors graph. We remark that the Random-Broadcast-Max converges faster than the two other algorithms which have similar convergence speeds. After a short *first hit* step, the Random-Walk is faster than the Random-Pairwise-Max for the first 150 iterations but slows down after that and is finally slower. This is due to the fact that Random-Walk informs a new sensor very frequently when only a few sensors are aware whereas the last sensors take a long time to inform as it is a *local* algorithm. In contrast, the Random-Pairwise-Max performs updates uniformly on the graph and is thus more successful when many sensors are informed.

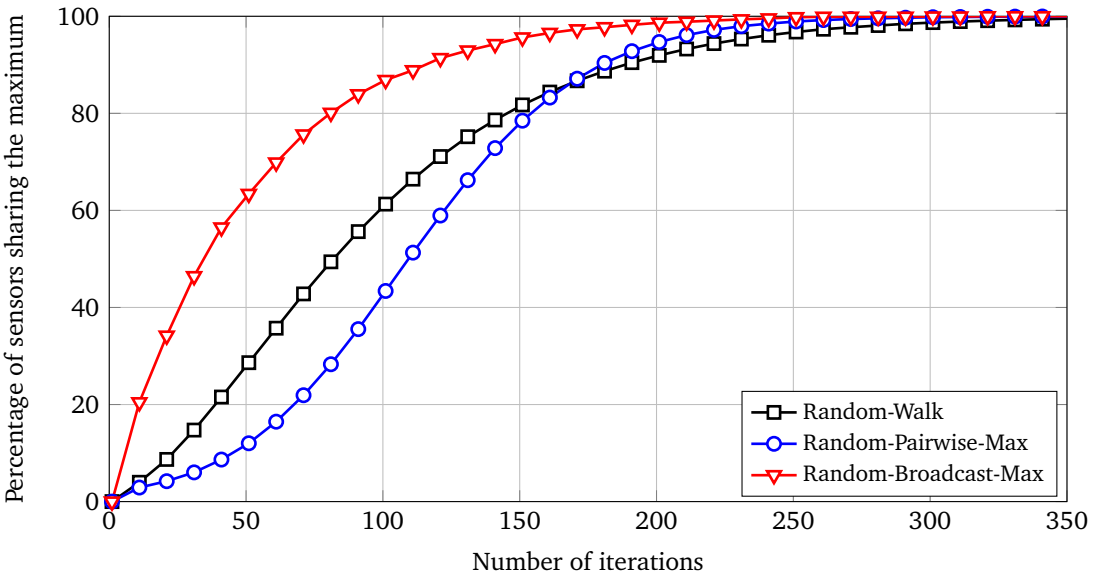


Figure 2.3: Percentage of informed nodes versus the number of iterations.

In Figure 2.4, we plot the (empirical) mean number of communications for reaching convergence and the associated upper-bounds (given by Theorems 2.4 and 2.7) for Random-Pairwise-Max and Random-Broadcast-Max versus the number of sensors N. The number of communications is a good indicator in the case of WSNs as the power constraints are important, it is simply obtained by multiplying the number of iterations by the number of communications per iteration, that is, 1 for the Random-Walk and the Random-Broadcast-Max[3] and 2 for the Random-Pairwise-Max. We again observe that the Random-Broadcast-Max outperforms the Random-Pairwise-Max; however, due to its communication cost the Random-Pairwise-Max is no longer faster than the Random-Walk. For small networks, the graph is almost always complete (see Fig. 2.2c) and thus our bound is tight as mentioned in Section 2.4.3-a for the Random-Broadcast-Max. When the network size increases, the upper-bounds become quite pessimistic due to the various used simplifications (in the case of Random-Pairwise-Max, we use the inequality of Eq. (2.3) and the approximation $d_i \approx 1/d_{\max}$; in the case of Random-

[3]we assume that a point-to-point communications costs roughly as much as broadcasting in a wireless setup.

Broadcast-Max, we rely on the spanning tree instead of the whole graph and we broadcast the information layer per layer). The looseness of our bounds is in fact inherent to the problem of rumor spreading on graphs as soon as one wants to consider all kinds of graphs; the graph-related tools and inequalities used in our derivations are indeed similar to the ones used in the rumor spreading literature so the derived bounds suffer from the same tightness problem.

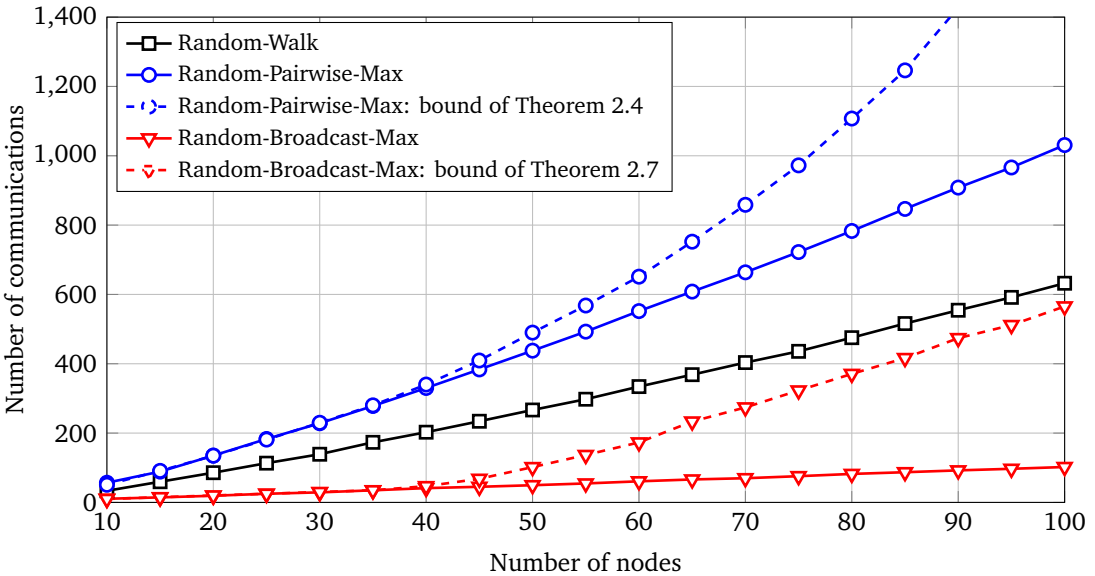


Figure 2.4: (Empirical) mean number of communications for reaching convergence and associated upper-bounds versus N.

As the Random-Broadcast-Max is much more interesting in terms of performance, we hereafter focus on it exclusively. In Figure 2.5, with $N = 40$ sensors, we plot the histogram of the convergence time and i) the quantile at $1/N$ (that is the value such that a fraction of $1/N$ of the trials are greater than this value); ii) the upper-bound of Theorem 2.8. Once again, our bound is loose which illustrates the fact that it is very hard to obtain tight bounds for spreading problems on general graphs.

2.6 Conclusion

In this chapter, we presented and analyzed the problem of distributed estimation of the maximal initial value in a WSN. After detailing the problem, we examined precisely the similitudes and differences between this problem and the well-known rumor spreading problem. Then, we presented and analyzed three algorithms: i) the Random-Walk which acts here as a toy example; ii) the Random-Pairwise-Max which mimics the well-known Random Gossip; and iii) the Random-Broadcast-Max which uses broadcast communications.

We showed that, roughly speaking, we pay a factor of the size of the network due to the fact that the sensors do not know if they have the wanted information (maximal value or

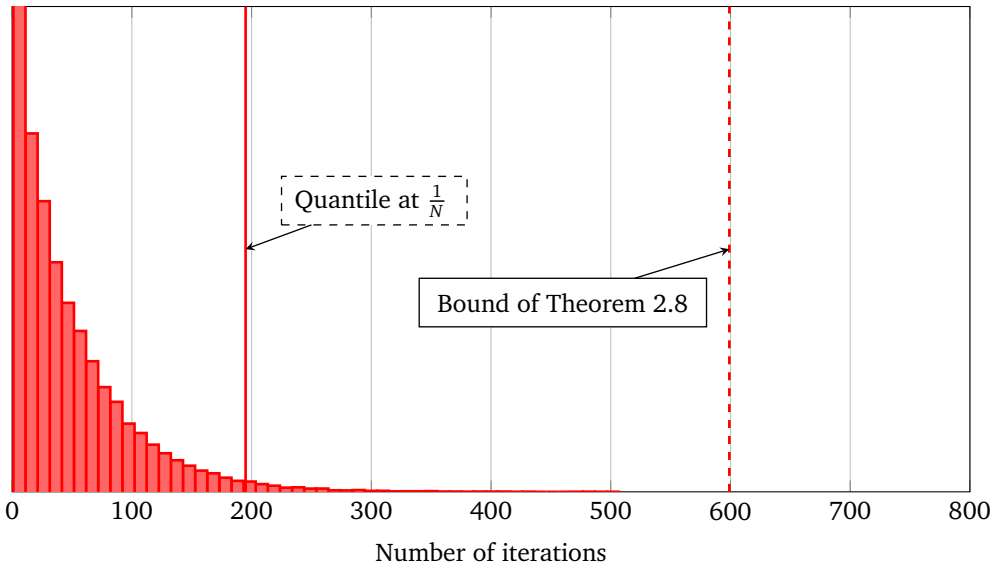


Figure 2.5: Histogram of the convergence time and associated upper-bound with probability $(1-1/N)$ for the Random-Broadcast-Max when $N = 40$.

rumor). We also observed that the use of the broadcast nature of the wireless channel improves dramatically the convergence speed of gossip algorithms for the maximum estimation.

In the next chapter, we will consider the problem of distributed estimation of the average value. Comforted in the idea that broadcasting would improve the convergence speed we will exhibit the convergence issues due to broadcasting in an averaging context and present a way to overcome this problem.

This work has led to the following publications:

J1 **F. Iutzeler**, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 6103–6107, November 2012.

C1 **F. Iutzeler**, J. Jakubowicz, W. Hachem, and P. Ciblat, "Distributed estimation of the maximum value over a Wireless Sensor Network," in *Asilomar Conference on Signals, Systems and Computers*, pp. 62–66, November 2011.

N1 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "Estimation distribuée du maximum dans un réseau de capteurs," in *Colloque GRETSI*, September 2011.

CHAPTER 3

DISTRIBUTED AVERAGE CONSENSUS

In this chapter, we will study the problem of distributed average consensus which consists in making a network to share the average of its initial values by performing local communications and linear operations at each sensor.

3.1 Introduction

As mentioned in Chapter 1, one of the most studied problems in WSNs is how to reach consensus over the average of the initial measurements of the sensors in an asynchronous way. In the seminal paper written by Boyd *et al.* [11], the problem is solved through the so-called *Random Gossip* algorithm in with at each iteration a randomly chosen sensor chooses one of its neighbors, then they exchange their values and update them by taking the mean between their received and former values. In the rest of the literature, the main proposed improvements for averaging algorithms (and any gossip algorithm in general) were based on either i) a better exploitation of the global geometry of the network; or ii) the use of broadcast communications.

The network structure was exploited in [37, 38] in order to perform a better mixing over the graph. Indeed, [37] proposed an algorithm called *Geographic Gossip* very similar to the random gossip but where the two exchanging nodes are not necessarily connected; the mixing is thus more global and hence the algorithm converges faster than the standard random gossip but this algorithm needs routing between the *waking* and the *chosen* sensor. This routing step can be in terms of communications/overhead and energy. [38] improved this algorithm by collecting the values of the sensors along the way of the routing, the chosen sensor thus has the values of all the sensors on the route between the waking sensor and itself. The chosen sensor can thus compute the average of these values and send it back through the same route as before. This algorithm named *Randomized Path Averaging* converges even faster than geographic gossip but still needs a costly routing. All these algorithms suffer from the need of costly routing. In addition, all the algorithms assume that the node receiving information feeds back its information to the sending node. This assumption can be very restrictive in

case of link failures (which may occur in real-life static network) or mobile wireless networks. Therefore we would like to propose feedback-free (and routing-free) gossip algorithms.

We have seen in the previous chapter that broadcast-based algorithms converge much faster than pairwise-based ones in the case of maximum value estimation. Thus it is natural to propose broadcast-based averaging algorithms. In the most simple algorithm of this type, called *Broadcast Gossip* [7], the waking node broadcasts its value to all its neighbors which update their estimates by taking the mean between their received and former values, this algorithm is in addition feedback-free, and so should be very promising. Indeed, it has a very fast convergence to a consensus but unfortunately does not converge to the true average of the initial values but to a random variable centered over the true average which is problematic and thus prevents its use. The inexact final value is due to the fact that the sum of the sensors estimates is not preserved over time. Some other broadcast-based averaging algorithms that converge to the true average exist but they need feedback. This feedback is a problem when link failures happen (as mentioned in previous paragraph) and also induces a non-negligible communication cost. In this context (broadcast with feedback-based algorithms), one can quote the *Neighborhood Gossip* [39] in which the waking sensor broadcasts a beacon to all its neighbors which then send back their values (the collisions are avoided by superposition coding) to the waking sensor which computes and broadcasts the average of the communicated values. The multiple transmissions and the multiple access issues are the negative points of this algorithm in the WSN setup. One can also quote the *Greedy Gossip with Eavesdropping* [40] which is a modified version of the random gossip where the waking sensor chooses its neighbor with the most different value to exchange, every sensor knows its neighbors values as the sensors broadcast their value after each update. As a conclusion, finding a feedback-free broadcast based algorithm converging to the true average is still an open issue. We propose hereafter to fix this issue.

To do so, we cannot rely on the standard gossip framework. Indeed, it is impossible to find such an algorithm via this framework as remarked in [41]. In contrast, as remarked in [42] [1], the so-called *Sum-Weight* framework introduced in [43] is well adapted to the feedback-free algorithms. We will also show that it is well adapted to design a (feedback-free) broadcast-based algorithm. In this framework, each sensor maintains and updates *two* local variables instead of one; the average estimate being the quotient of these two variables. In opposition to the two different variables of the Sum-Weight framework, we will call the standard gossip algorithms depending on only one variable *single-variate*. The Sum-Weight formalism was originally proposed and studied in terms of convergence for synchronous average computation, the asynchronous case was treated by Bénézit *et al.*[42] and later in [41]. However, the convergence speed has never been theoretically evaluated except in [43] for a very specific case.

As a conclusion, our objective is twofold: on the one hand, we propose to theoretically

[1] where the authors generalize the randomized path averaging to a new feedback-free algorithm leading to the so-called *One-Way Path Averaging* algorithm

analyze the convergence speed of any algorithm based on the Sum-Weight framework; on the other hand, we design a feedback-free broadcast-based algorithm using this framework which converges very quickly and outperforms existing algorithms.

The chapter is organized as follows: in Section 3.2 we remind some results and algorithms associated with the standard (single-variate) framework. In Section 3.3, we recall the Sum-Weight framework and introduce our assumptions. In Section 3.4, we provide our results about the convergence and convergence speed for sum-weight-based algorithms. This section corresponds to the main contributions of the chapter. In Section 3.5, we propose a new feedback-free broadcast-based Sum-Weight algorithm called BWGossip. In Section 3.6, we revisit the results on the standard (single-variate) gossip algorithms according to results obtained in Section 3.4. Finally, in Section 3.7, we provide one application for averaging algorithms which is the distributed spectrum sensing in a cognitive radio context. We especially see that the BWGossip helps us to fix some problems raised in this Section.

3.2 Standard (single-variate) framework

3.2.1 Model

According to Chapter 1, we consider a WSN modeled by a graph $\mathscr{G} = (V, E)$ and each sensor i has an initial value x_i^0 but now the goal of the algorithms is to compute $x_{\text{ave}} \triangleq 1/N \sum_{i=1}^{N} x_i^0$ by i) exchanging only locally in the sense of Assumption 1.1; and ii) through linear operations with non-negative coefficients[2]. In the standard single-variate framework, all the algorithms can then be expressed with a matrix formulation as

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} x^k \tag{3.1}$$

where the process $\{\mathbf{K}_{\xi^k}\}_{k>0}$, valued in the set $\mathscr{K} = \{\mathbf{K}_i\}_{i=1,\dots,M}$, is i.i.d.. The set $\mathscr{K}$ contains M non-negative matrices whose support is included in the support of $\mathbf{I} + \mathbf{A}$ with $\mathbf{A}$ the adjacency matrix of $\mathscr{G}$. The goal of an averaging algorithm can be written

$$x^k \rightsquigarrow x_{\text{ave}} \mathbb{1} \ (= \frac{1}{N} \mathbb{1}\mathbb{1}^{\mathrm{T}} x^0 = \mathbf{J} x^0) \tag{3.2}$$

where $\rightsquigarrow$ is a convergence to be defined later.

Before going further, it seems natural that the matrices of $\mathscr{K}$ satisfy two properties:

- **Sum conservation:** in order to keep the sought information (the average/sum of the initial values) throughout the processing time, it is mandatory that the sum of the nodes variables should be unchanged at each iteration and so identical to the sum of the initial values given by x_{ave}. Therefore, we need

$$\mathbb{1}^{\mathrm{T}} x^{k+1} = \mathbb{1}^{\mathrm{T}} x^k. \tag{3.3}$$

  As $x^k = \mathbf{K}_{\xi^k} \mathbf{K}_{\xi^{k-1}} \dots \mathbf{K}_{\xi^1} x^0$, Eq. (3.3) holds if and only if each matrix of the set $\mathscr{K}$ is **column-stochastic**.

[2] this is actually not restrictive as it seems logical to combine receive values using weighted means.

- **Consensus conservation:** it seems important that the consensus is stable (if it exists). Let c be the consensus value. The stability leads to

$$x^k = c\mathbb{1} \Rightarrow x^{k+1} = c\mathbb{1} \tag{3.4}$$

As $x^{k+1} = \mathbf{K}_{\xi^{k+1}} x^k$, Eq. (3.4) for any c holds if and only if the set $\mathscr{K}$ is **row-stochastic**. Notice that, in the remainder of this Section, the relative importance of both above-mentioned properties will be analyzed. Therefore we will not assume these properties by default but case by case.

Finally, it seems natural that a sensor always keeps a part of its own value at each iteration. This implies that each matrix of the set $\mathscr{K}$ has positive diagonal entries. Therefore, we will consider that the following Assumption 3.1 holds throughout this Chapter.

Assumption 3.1. *The matrices of $\mathscr{K}$ are non-negative and positive diagonal entries.*

This section is organized as follows: in Subsection 3.2.2, we consider the doubly-stochastic matrices case whereas in Subsection 3.2.3, we focus on the strictly non-doubly-stochastic (either row or column) matrices case.

3.2.2 Case of doubly-stochastic matrices

In order to introduce the problem in a more simple way, we start this Subsection by focusing on the synchronous communications context (even if the contributions of the thesis concern the asynchronous communications context) in Section 3.2.2-a.

3.2.2-a Synchronous communications context

In synchronous communications context, all the nodes speak at the time and do always the same operation. We consider that collisions are avoided by choosing an appropriate multiple access scheme. Consequently, the algorithms are no random and simply characterize by the initial vector x^0 and a unique $\mathbf{K}$ (satisfying Assumption 3.1 and the doubly-stochastic property, obviously). We so have

$$x^{k+1} = \mathbf{K}x^k. \tag{3.5}$$

Notice that the below-mentioned existing results are based on [44, 45, 3, 46] and [47].

According to Chapter 1, $\mathbf{K}$ can be viewed a transition probability matrix of a homogeneous Markov Chain since it is row-stochastic. From [45, Prop. 8.3] (or [3, Theorem 8.5.1]), this related homogeneous Markov chain is ergodic if $\mathbf{K}$ is primitive. Thanks to Definition 1.10, we know that, if $\mathbf{K}$ is primitive and so ergodic, then it exists a positive vector v (with $\mathbb{1}^{\mathrm{T}} v = 1$) such that

$$\lim_{m\to\infty} (\mathbf{K})^m = \mathbb{1} v^{\mathrm{T}}.$$

As $\mathbf{K}$ is also column-stochastic, we have $v = (1/N)\mathbb{1}$ and so

$$\lim_{m\to\infty} (\mathbf{K})^m = \frac{1}{N}\mathbb{1}\mathbb{1}^{\mathrm{T}}. \tag{3.6}$$

Eq (3.6) implies directly that

$$\lim_{k_t o \infty} x^k = x_{\text{ave}} \mathbb{1}$$

which corresponds to the convergence proof.

Once the convergence ensures, we would like to analyze the convergence speed to the consensus. Before going further, we need two preliminary (well-known) propositions. For pedagogical reasons, we will also remind their proofs.

Result 3.2. *Under Assumption 3.1, if* $\mathbf{K}$ *is primitive, then* $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ *is also primitive.*

Proof. As $\mathbf{K}$ is non-negative, we have $(\mathbf{K}^{\mathrm{T}}\mathbf{K})_{i,j} \geq \mathbf{K}_{i,i}\mathbf{K}_{i,j}$. Furthermore, as $\mathbf{K}$ has positive diagonal entries, we get that the support of $\mathbf{K}$ is included in the support of $\mathbf{K}^{\mathrm{T}}\mathbf{K}$. This implies that if $\mathscr{G}(\mathbf{K})$ is strongly connected (and so $\mathbf{K}$ primitive as it is irreducible with positive diagonal entries), then $\mathscr{G}(\mathbf{K}^{\mathrm{T}}\mathbf{K})$ is also strongly connected and so $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ is irreducible with positive diagonal entries and so primitive. □

Result 3.3. *Under Assumption 3.1, if* $\mathbf{K}$ *is doubly-stochastic and primitive, then*

$$\rho\left(\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{J}\right) < 1. \tag{3.7}$$

Proof. As $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ is doubly-stochastic, its largest eigenvalue in magnitude is 1 associated with left- and right-eigenvectors $1/\sqrt{N}\mathbb{1}$. According to Result 3.2, $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ is primitive. So thanks to Theorem 1.5, 1 is the sole eigenvalue of magnitude 1. Consequently, all the eigenvalue of $\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{J}$ are strictly less than 1 in magnitude which concludes the proof. □

Eq. (3.7), will play a great role for finding the convergence speed. Indeed, the term x^k can be decomposed as follows

$$x^k = \mathbf{J}x^k + \mathbf{J}^{\perp}x^k$$

and as $\mathbf{K}$ is column-stochastic, one can easily see that $\mathbf{J}x^k = x_{\text{ave}}\mathbb{1}$ for any k. Consequently, we have

$$x^k = x_{\text{ave}}\mathbb{1} + \mathbf{J}^{\perp}x^k \quad \text{(due to the column-stochasticity of } \mathbf{K}) \tag{3.8}$$

which implies the convergence of the $\{x^k\}_{k>0}$ to the average consensus is equivalent to the convergence of $\|\mathbf{J}^{\perp}x^k\|_2^2$ to 0. As $\mathbf{K}$ is row-stochastic, Proposition 1.7 holds and so we have

$$\|\mathbf{J}^{\perp}x^{k+1}\|_2^2 = \|\mathbf{J}^{\perp}\mathbf{K}(\mathbf{J}^{\perp}x^k)\|_2^2 \quad \text{(due to the row-stochasticity of } \mathbf{K}).$$

Consequently, we have a simple link between the projections of the nodes variables into $\text{span}(\mathbb{1})^{\perp}$ at time k and $(k+1)$ enabling us to conclude quickly. Indeed, thanks to the previous equation, we have

$$\|\mathbf{J}^{\perp}x^{k+1}\|_2^2 \leq |||\mathbf{J}^{\perp}\mathbf{K}|||_2^2 \|\mathbf{J}^{\perp}x^k\|_2^2.$$

After simple algebraic manipulations, if $\mathbf{K}$ is column-stochastic (cf. Proposition 1.8), we have

$$|||\mathbf{J}^{\perp}\mathbf{K}|||_2^2 = \rho\left(\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{J}\right)$$

which is less than 1 according to Result 3.3.

To sum up, we have the following well-known Theorem.

Theorem 3.4. *Let Assumption 3.1 hold. If* $\mathbf{K}$ *is doubly-stochastic and primitive, the sequence* $\{x^k\}_{k>0}$ *defined in Eq.* (3.5)

- *converges to* $x_{ave}\mathbb{1}$.
- *the squared error* $\|x^k - x_{ave}\mathbb{1}\|_2^2$ *goes exponentially to zero with the slope* $\rho\left(\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{J}\right)$ *which is ensured to be less than* 1.

Example: Metropolis algorithm. Each node replaces its own value with a weighted average of its previous value and the values of all its neighbors at each iteration. The *Metropolis weights*, introduced in the context of Markov chain Monte Carlo [48, 49], are as follows

$$\mathbf{K}_{i,j} = \begin{cases} \frac{1}{1+\max\{d_i,d_j\}} & \text{if } j \in \mathscr{N}_i \\ 1 - \sum_{j\in\mathscr{N}_i} \mathbf{K}_{i,j} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

In order for the algorithm to converge, we would like to check the sufficient conditions on the matrix $\mathbf{K}$ given in Theorem 3.4. The above-defined matrix $\mathbf{K}$ trivially satisfies Assumption 3.1 and is doubly-stochastic. We now just have to check its primitivity. This matrix has the same support as $\mathbf{I}+\mathbf{A}$ where $\mathbf{A}$ is the adjacency matrix of the underlying (undirected, self-loop-free) communication graph $\mathscr{G}$. According to Proposition 1.2, if $\mathscr{G}$ is (strongly) connected, then $\mathbf{A}$ is irreducible, so the matrix $\mathbf{I}+\mathbf{A}$ is primitive and so the support $\mathbf{I}+\mathbf{A}$ is primitive too which implies that $\mathbf{K}$ is primitive. So this algorithm converges to the true consensus as soon as the communication graph is connected.

We are now ready to move on the asynchronous communications context in Section 3.2.2-b.

3.2.2-b Asynchronous communications context

Now, the sequence $\{x^k\}_{k>0}$ is defined as in Eq. (3.1). We would like to obtain conditions on the set $\mathscr{K}$ for converging to the true consensus almost surely. To do so, we have to see the choice of the matrices of $\mathscr{K}$ at each time as a random process. Obviously, the random may differ from the considered algorithm. We will just consider that this process is i.i.d. for the sake of simplicity. In addition, we consider that each matrix of $\mathscr{K}$ satisfies Assumption 3.1 and are doubly-stochastic. The results will be similar to the synchronous case by adding some mathematical expectations well localized!

Notice that the below-mentioned existing results are based on [6, 42, 50] and [8]. We still consider that Assumption 3.1 holds.

Once again, one can use the Markov Chain formalism to prove the convergence to a consensus (here in the almost sure sense). The sequence of random matrices $\{\mathbf{K}_{\xi^k}\}_{k>0}$ is assumed i.i.d.. We obviously have

$$x^k = \mathbf{P}^{1,k}x^0$$

with

$$\mathbf{P}^{s,k} \triangleq \mathbf{K}_{\xi^k}\mathbf{K}_{\xi^{k-1}}\ldots\mathbf{K}_{\xi^s} \quad \text{for any } s \text{ and } k \geq s. \tag{3.10}$$

Clearly, x^k is obtained through a (backward) concatenation of transition probability matrix of an heterogeneous Markov Chain since each matrix of the set $\mathscr{K}$ is row-stochastic. According to Section 1.2.5-b, when backward direction is considered, the notions of weak ergodicity and strong ergodicity are equivalent. In [42, Lemma 4.3], it is proven that, if $\mathbb{E}[\mathbf{K}]$ is primitive, then the sequence $\{\mathbf{K}_{\xi^k}\}_{k>0}$ is weak ergodic (in backward direction) almost surely and so strong ergodic. Consequently, according to Definition 1.10, it exists one non-negative vector v with unit sum such that $\lim_{k\to\infty}\mathbf{P}^{1,k}=\mathbb{1}v^{\mathrm{T}}$. In addition, as the matrices in the set $\mathscr{K}$ are also column-stochastic, we have $v=(1/N)\mathbb{1}$. Finally, as each matrix in $\mathscr{K}$ is doubly-stochastic and as $\mathbb{E}[\mathbf{K}]$ is primitive, we obtain that

$$x^k = x_{\text{ave}}\mathbb{1} \quad \text{almost surely}$$

So the condition of primitivity has to be checked by the mean matrix instead of by each matrix in $\mathscr{K}$ at each time.

We hereafter would like to inspect the convergence speed. Once again, the column-stochasticity of each matrix in $\mathscr{K}$ implies that we only have to analyze the convergence speed of $\|\mathbf{J}^{\perp}x^k\|^2$. Using the same approach as in the synchronous case, we have, due to the row-stochasticity (cf. Proposition 1.7),

$$\|\mathbf{J}^{\perp}x^{k+1}\|_2^2 = \|(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})\mathbf{J}^{\perp}x^k\|_2^2 = (\mathbf{J}^{\perp}x^k)^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})(\mathbf{J}^{\perp}x^k). \tag{3.11}$$

Taking the expectation over ξ^{k+1}, we get

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{J}^{\perp}x^{k+1}\|_2^2|x^k\right] &= (\mathbf{J}^{\perp}x^k)^{\mathrm{T}}\mathbb{E}[(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})](\mathbf{J}^{\perp}x^k)\\ &\leq \rho\left(\mathbb{E}\left[(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})\right]\right)\|\mathbf{J}^{\perp}x^k\|_2^2. \end{aligned}\tag{3.12}$$

Now, the main important term providing the slope of the exponential decrease is

$$\rho\left(\mathbb{E}\left[(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_{\xi^{k+1}})\right]\right)$$

and we have to prove that it is less than 1. As each matrix in $\mathscr{K}$ is also column-stochastic, we get that for every $i\in\{1,...,M\}$,

$$(\mathbf{J}^{\perp}\mathbf{K}_i)^{\mathrm{T}}(\mathbf{J}^{\perp}\mathbf{K}_i) = \mathbf{K}_i^{\mathrm{T}}\mathbf{K}_i - \mathbf{J}$$

which means that we just have to verify that $\rho(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]-\mathbf{J})$ is less than 1. As $\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]$ is trivially doubly-stochastic, we have $\rho(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}])=1$. Similarly to Result 3.2 (so by considering the associated support matrices), we have that Assumption 3.1 leads to the primitivity of $\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]$ if $\mathbb{E}[\mathbf{K}]$ is primitive. Consequently, by Perron-Frobenius Theorem, 1 is the unique eigenvalue of maximum modulus, and $\mathbb{1}$ is the unique eigenvector of $\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]$ associated with the eigenvalue 1. So the eigenvalues of $\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]-\mathbf{J}$ are strictly less than 1 since the matrix $\mathbf{J}$ has removed the eigenspace associated with $\mathbb{1}$ in $\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]$. Finally, the primitivity of $\mathbb{E}[K]$ leads to

$$\rho\left(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]-\mathbf{J}\right)<1. \tag{3.13}$$

We just have to check now that Eqs. (3.12)-(3.13) enable us to fix the problem. For any $\varepsilon > 0$, thanks to Markov's inequality, we have

$$\sum_{k=0}^{\infty} \mathbb{P}[\|\mathbf{J}^{\perp} x^k\|^2 > \varepsilon] \leq \frac{1}{\varepsilon} \nu \sum_{k=0}^{\infty} \mathbb{E}[\|\mathbf{J}^{\perp} x^k\|^2]$$

According to Eq. (3.12), we obtain that

$$\sum_{k=0}^{\infty} \mathbb{P}[\|\mathbf{J}^{\perp} x^k\|^2 > \varepsilon] \leq \frac{C}{\varepsilon} \sum_{k=0}^{\infty} r^k$$

with $r = \rho\left(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}] - \mathbf{J}\right)$ and $C = \|\mathbf{J}^{\perp} x^0\|^2$. According to Eq. (3.13), we have $r < 1$, so

$$\sum_{k=0}^{\infty} \mathbb{P}[\|\mathbf{J}^{\perp} x^k\|^2 > \varepsilon] \leq \frac{C}{\varepsilon(1-r)} < \infty,$$

and the Borel-Cantelli Lemma holds which implies $\|\mathbf{J}^{\perp} x^k\|^2$ converges to zero almost surely at an exponential rate equal to r. To sum up, we have the following Theorem.

Theorem 3.5. *Let Assumption 3.1 hold. If each matrix in $\mathscr{K}$ is doubly-stochastic and i.i.d. and if $\mathbb{E}[\mathbf{K}]$ primitive, the sequence $\{x^k\}_{k>0}$ defined in Eq.* (3.1)

- *converges to $x_{ave}\mathbb{1}$ almost surely.*
- *the mean squared error $\mathbb{E}[\|x^k - x_{ave}\mathbb{1}\|_2^2]$ goes exponentially to zero with the following slope $\rho\left(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}] - \mathbf{J}\right)$ which is ensured to be less than* 1.

Example: Random Gossip algorithm. The *Random Gossip* algorithm, introduced in [11], relies on a pairwise exchange between the activating node and one of its neighbors chosen uniformly. Then they compute the average of their received and former values. The algorithm is summed up in next Table.

Random Gossip

Let i be the node activating at time k.

- ► i chooses a neighbor j uniformly in $\mathscr{N}_i$ and they exchange their values.
- ► Both i and j then update: $x_i^{k+1} = x_j^{k+1} = \frac{x_i^k + x_j^k}{2}$.

As mentioned in Section 1.1.2, the agents activate through an i.i.d. process and for any time $k > 0$ and any node $i \in V$, $\mathbb{P}[i \text{ activates at time } k] = 1/N$.

Using the matrix framework described in previous Sections, the iteration at time k can be written as follows

$$x^{k+1} = \mathbf{K}_{\{i,j\}} x^k$$

with

$$\mathbf{K}_{\{i,j\}} \triangleq \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & 1/2 & & & & 1/2 & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & 1/2 & & & & 1/2 & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix}.$$

According to our framework, we have $\mathscr{K} = \{\mathbf{K}_{\{i,j\}}\}_{\{i,j\}\in E}$ and $\xi^k = \{i, j\}$ with probability $N^{-1}(|\mathscr{N}_i|^{-1} + |\mathscr{N}_j|^{-1})$.

In order to apply results developed previously, we just have to check that matrices in $\mathscr{K}$ satisfy all the constraints. Assumption 3.1 holds since each matrix in $\mathscr{K}$ is non-negative with a positive diagonal. In addition, it is doubly-stochastic. Thus, we now just have to check that $\mathbb{E}[\mathbf{K}]$ is primitive for applying Theorem 3.5. As this matrix has the same support as $\mathbf{I}+\mathbf{A}$ where $\mathbf{A}$ is the adjacency matrix of the underlying communication graph $\mathscr{G}$, $\mathbb{E}[\mathbf{K}]$ is primitive as soon as $\mathscr{G}$ is connected (as done for Metropolis algorithm example.).

3.2.3 Case of non-doubly-stochastic matrices

First of all, we will see that the double stochasticity of the updates requires a feedback. Therefore, when feedback-free algorithm is of interest, the update matrices will be either row-stochastic or column-stochastic but not both simultaneously. Hereafter, we will inspect existing results under this less-restrictive assumption.

Let us focus on the relationship between doubly-stochasticity and the feedback requirement. Let $\mathbf{K}$ be an update matrix. If the node i provides information to the node j, then $\mathbf{K}_{j,i} > 0$. Node j performs weight mean with these values of interest (in this toy example, its

own value and that of node i). We thus have

$$\mathbf{K} = \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & 1 & & & & & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & 1/2 & & & & & 1/2 & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix}$$

and, due to the weight mean, the sum of each row is equal to 1. By construction, the matrix $\mathbf{K}$ is row-stochastic *but* not column-stochastic. In order to be doubly-stochastic, the i-th row must modify entries (i,i) and (i,j) and force them to be equal to $1/2$ (leading actually to the so-called Random Gossip algorithm described in Section 3.2.2-b). Having a non-null coefficient at entry (i,j) means that j must also provides its value to i. Therefore, the double stochasticity property requires feedback communications.

3.2.3-a Row-stochastic matrices

The previous example shows that it is easy to design a lot of algorithms by considering different weight means at the receive node. Then such algorithms have row-stochastic update matrices. Therefore, we first focus on this case of row-stochastic matrices.

As matrices in $\mathscr{K}$ are row-stochastic, they can still be seen as transition probability matrices of an heterogeneous Markov Chain but in backward direction. According to Section 1.2.5-b, weak and strong ergodicity are equivalent. Moreover, like [42, Lemma 4.3], if the update matrices satisfy Assumption 3.1 and $\mathbb{E}[\mathbf{K}]$ is primitive, then the product of matrices in $\mathscr{K}$ chosen in an i.i.d. manner, is weak ergodic and so strong ergodic almost surely which implies that Eq. (1.3) applies (we just have to consider the backward direction instead of the forward one.).

So let $\xi = \{\xi^k\}_{k\in\mathbb{N}}$ be a realization of the chosen node. According to Definition 1.10, it exists almost surely a non-negative vector v (with $\mathbb{1}^{\mathrm{T}}v = 1$ and depending on ξ) such that

$$\lim_{k\to\infty} \mathbf{K}_{\xi^k} \cdots \mathbf{K}_{\xi^1} \to \mathbb{1}v^{\mathrm{T}}.$$

Given the previous equation, we have that

$$x^k \overset{k\to\infty}{\to} (v^{\mathrm{T}}x^0)\mathbb{1} \quad \text{(almost surely)}$$

Moreover as matrices in $\mathscr{K}$ are not column-stochastic,we have $v \neq (1/N)\mathbb{1}$. As a conclusion, the algorithm will converge to a consensus but the consensus is random and different from x_{ave}.

Nevertheless, notice that if $\mathbb{E}[\mathbf{K}]$ is column-stochastic (so column stochastic in expectation not for each realization), then one can prove $\mathbb{E}[v] = (1/N)\mathbb{1}$ which implies that the algorithm converges to an unbiased random consensus [7, 41]. We sum up these results in the following Theorem.

Theorem 3.6. *Let Assumption 3.1 hold. If each matrix in $\mathscr{K}$ is row-stochastic and i.i.d. and if $\mathbb{E}[\mathbf{K}]$ primitive, there is a non-negative random vector v with ($\mathbb{1}^{\mathrm{T}}v = 1$) such that the sequence $\{x^k\}_{k>0}$ defined in Eq.* (3.1)

- *converges to $(v^{\mathrm{T}}x^0)\mathbb{1}$ almost surely,*
- *Furthermore, if $\mathbb{E}[\mathbf{K}]$ is column-stochastic, then $\mathbb{E}[v^{\mathrm{T}}x^0] = x_{ave}$ and the algorithm is unbiased.*

As a remark, the row-stochasticity implies that the consensus is stable but does not ensure that the sought main information (here, the average) is kept. Consequently, it is hopeless to expect the convergence to the true value since the true value (here, the average) is lost as soon as the first iteration. In order to design algorithms converging to the true consensus, the column-stochasticity is much more important since it ensures the average conservation at each iteration. Therefore, we now focus on this case. Before going further, let us give an example of a (feedback-free broadcast) algorithm based on row-stochastic update matrices.

Example: Broadcast Gossip algorithm. The *Broadcast Gossip* algorithm, introduced in [7], relies on a broadcast of its value by the activating node to its neighbors. Then its neighbors compute the average of their received and former values. This algorithm is clearly feedback-free as wanted. The algorithm is summed up in next Table.

Broadcast Gossip

Let i be the activating node at time k.

- ► i broadcasts its estimate to all its neighbors.
- ► Every neighbor $j \in \mathscr{N}_i$ updates: $x_j^{k+1} = \frac{x_i^k + x_j^k}{2}$.

As mentioned in Section 1.1.2, the agents activate through an i.i.d. process and for any time $k > 0$ and any node $i \in V$, $\mathbb{P}[i \text{ activates at time } k] = 1/N$.

Using the matrix framework described in previous Sections, the iteration at time k can be written as follows

$$x^{k+1} = \mathbf{K}_i x^k$$

with

$$\mathbf{K}_i \triangleq \begin{bmatrix} 1/2 & & & & 1/2 & & & & \\ & 1/2 & & & 1/2 & & & & \\ & & 1 & & & & & & \\ & & & \ddots & & & & & \\ & & & & 1 & & & & \\ & & & & 1/2 & 1/2 & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \\ & & & & 1/2 & & & & 1/2 \end{bmatrix}.$$

According to our framework, we have $\mathscr{K} = \{\mathbf{K}_i\}_{i \in V}$ and $\xi^k = i$ with probability $1/N$.

In order to apply the previously developed results, we just have to check that matrices in $\mathscr{K}$ satisfy all the constraints. Assumption 3.1 holds since each matrix in $\mathscr{K}$ is non-negative with a positive diagonal. In addition, it is only row-stochastic. Thus, we now just have to check that $\mathbb{E}[\mathbf{K}]$ is primitive for applying Theorem 3.6. As this matrix has the same support as $\mathbf{I} + \mathbf{A}$ where $\mathbf{A}$ is the adjacency matrix of the underlying communication graph $\mathscr{G}$, $\mathbb{E}[\mathbf{K}]$ is primitive as soon as $\mathscr{G}$ is connected (as done for Metropolis algorithm example.). Moreover, the sum of the i-th column of $\mathbb{E}[\mathbf{K}]$ is equal to $1 + 1/N(d_i/2) - d_i 1/N(1/2) = 1$. So, $\mathbb{E}[\mathbf{K}]$ is column-stochastic and the consensus value is well centered on x_{ave} as proven in [7, 41].

3.2.3-b Column-stochastic matrices

Let us now focus on the case of column-stochastic update matrices. Compared to the row-stochastic update matrices case, the average is kept at each iteration. We just want to exhibit algorithms providing this average.

As the update matrices are column-stochastic, they can not be viewed as transition probability matrices of Markov Chain. It is easy to overcome this problem by working on

$$(\mathbf{K}_{\xi^k} \cdots \mathbf{K}_{\xi^1})^{\mathrm{T}} = \mathbf{K}_{\xi^1}^{\mathrm{T}} \cdots \mathbf{K}_{\xi^k}^{\mathrm{T}} \tag{3.14}$$

instead on

$$\mathbf{K}_{\xi^k} \cdots \mathbf{K}_{\xi^1}$$

as naturally done by the algorithm.

So we can work with a concatenation of row-stochastic "update matrices" (actually, the transpose of the real update matrices) but in forward direction. In forward direction case, weak and strong ergodicity are distinguished. Under similar assumptions as before, we can just ensure the weak ergodicity. This means that it exists a random sequence of non-negative vectors v^k (with $\mathbb{1}^{\mathrm{T}} v^k = 1$) such that

$$(\mathbf{K}_{\xi^k} \cdots \mathbf{K}_{\xi^1})^{\mathrm{T}} \sim \mathbb{1} {v^k}^{\mathrm{T}}$$

or equivalently

$$\mathbf{K}_{\xi^k} \cdots \mathbf{K}_{\xi^1} \sim v^k \mathbb{1}^{\mathrm{T}}. \tag{3.15}$$

Compared to the row-stochastic case, we have two fundamental differences: i) the vector v^k is moving so there is no stable consensus (this is logical since the consensus conservation related to the row-stochasticity is not assumed), ii) the vector v^k multiplies $\mathbb{1}$ by the left (and not by the right in the row-stochastic case). The last difference is crucial and will be the key way to solve our problem.

Let us inspect the consequence on x^k. We thus have that

$$x^k \sim v^k N x_{\text{ave}}$$

or equivalently, at node i, we have

$$x_i^k \sim v_i^k N x_{\text{ave}}. \tag{3.16}$$

We observe that the average is kept but is hidden in x_i^k by the unknown term v_i^k. Consequently the algorithms based on column-stochastic update matrices do not have any chance to converge to the true value even if this value is there !

To fix this issue, we have to remove the term v_i^k, and so we need side information on this term. Therefore, an other variable linked to the value of v_i^k has to be computed in parallel. But which one ? the answer will be given in next Section by the so-called Sum-Weight framework. The *Sum* corresponds to Eq. (3.16), and the *Weight* corresponds to the variable related to v_i^k.

3.3 Sum-Weight framework

According to Section 3.2.3-b, it seems reasonable to design algorithms with column-stochastic matrices, but then two variables are needed. Clearly, a way to compute v_i^k at node i in parallel with $v_i^k N x_{\text{ave}}$ is to apply Eq. (3.16) with an other initialization point, *i.e.*, $\mathbb{1}$. Therefore at each iteration, we will update two variables s^k (the sum as initialized with x^0) and w^k (the weight as initialized with $\mathbb{1}$) with the *same* update matrix. In such a case, the first (resp. second) variable behaves as $v_i^K N x_{\text{ave}}$ (resp. $v_i^K N$) at node i. Doing the division (assuming non-null v_i^k) leads to the average x_{ave}.

The Sum-Weight framework, introduced in [43] and adapted in [42] to the wireless network, is thus based on the joint update of two variables per node as above explained. Mathematically, we have two variables s_i^k and w_i^k at node i and time k.

We initialize with $s^0 = x^0$ and $w^0 = \mathbb{1}$. Then at time k, we have

$$\begin{aligned} s^{k+1} &= \mathbf{K}_{\xi^{k+1}} s^k \\ w^{k+1} &= \mathbf{K}_{\xi^{k+1}} w^k \\ x^{k+1} &= \tfrac{s^{k+1}}{w^{k+1}} \end{aligned} \tag{3.17}$$

where

- the division is elementwise,

- the process of update matrices $\{\mathbf{K}_{\xi^k}\}_{k>0}$ belonging to $\mathscr{K} = \{\mathbf{K}_i\}_{i=1,\dots,M}$ is i.i.d.,
- the set $\mathscr{K}$ contains M non-negative matrices whose support is included in the support of $\mathbf{I}+\mathbf{A}$ with $\mathbf{A}$ the adjacency matrix of $\mathscr{G}$.

The goal of Sum-Weight averaging algorithms is the same as the one of standard gossip algorithms (see Eq. (3.2)) and writes as follows

$$x^k \to x_{\text{ave}}\mathbb{1} \text{ almost surely.} \tag{3.18}$$

In the remainder of this Section, we will prove new results about the convergence and convergence speed of Sum-Weight-based averaging algorithms under the following assumptions.

Assumption 3.7. *The update matrices must verify:*

a. the matrices of $\mathscr{K}$ are non-negative, column-stochastic, and have positive diagonal entries;
b. the update matrices are chosen through an independent and identically distributed process $\{\xi^k\}_{k>0}$ valued in $\{1,\dots,M\}$;
c. $\mathbb{E}[\mathbf{K}]$ is primitive.

The column-stochasticity in Assumption 3.7a ensures that, for any k, $\sum_{i\in V} s_i^k = \sum_{i\in V} x_i^0$ and $\sum_{i\in V} w_i^k = N$. Notice that Assumption 3.7c holds if $\text{Supp}(\mathbb{E}[\mathbf{K}]) = (\mathbf{I}+\mathbf{A})$ and $\mathbf{A}$ is the adjacency matrix of a (strongly) connected graph.

3.4 Convergence of Sum-Weight-based averaging algorithms

The convergence proofs and convergence speed bounds derived in this Section correspond to our main contribution in distributed average consensus algorithms.

3.4.1 Preliminary results

In order to intuitively get how to prove the convergence x^k to the average consensus[3], let us write it as follows:

$$\begin{aligned} x^k &= \frac{s^k}{w^k} = \frac{\mathbf{P}^{1,k}x^0}{\mathbf{P}^{1,k}\mathbb{1}} \\ &= \frac{\mathbf{P}^{1,k}\mathbf{J}x^0}{w^k} + \frac{\mathbf{P}^{1,k}\mathbf{J}^\perp x^0}{w^k} \\ &= \frac{x_{\text{ave}}\mathbf{P}^{1,k}\mathbb{1}}{\mathbf{P}^{1,k}\mathbb{1}} + \frac{\mathbf{P}^{1,k}\mathbf{J}^\perp x^0}{w^k} \\ &= x_{\text{ave}}\mathbb{1} + \frac{\mathbf{P}^{1,k}\mathbf{J}^\perp x^0}{w^k}. \end{aligned} \tag{3.19}$$

[3] notice that we can not follow the approach done for standard (single-variate) algorithms since the update matrices are not row-stochastic. This especially implies that Eq. (3.11) does not hold anymore and the recursion on $\mathbf{J}^\perp x^k$ does not work anymore.

The last inequality indicates that proving that the convergence of x^k to $x_{\text{ave}}\mathbb{1}$ is equivalent to proving that $\mathbf{P}^{1,k}\mathbf{J}^{\perp}x^0/w^k$ vanishes. As it is uneasy to work with elementwise division, the proof has to be divided into two parts: i) proving that w^k is bounded away from zero; and ii) proving that $\mathbf{P}^{1,k}\mathbf{J}^{\perp}x^0$ vanishes for any x^0. This approach for analyzing the convergence of Sum-Weight algorithms is inspired by [43] (with a number of important differences explained below).

Formally, let us upper-bound the Squared Error (SE) by a product of two terms as follows

$$\begin{aligned} \|x^k - x_{\text{ave}}\mathbb{1}\|_2^2 &= \sum_{i=1}^{N} \left|x_i^k - x_{\text{ave}}\right|^2 = \sum_{i=1}^{N} \frac{1}{(w_i^k)^2} \left|s_i^k - x_{\text{ave}} w_i^k\right|^2 \qquad (3.20) \\ &= \sum_{i=1}^{N} \frac{1}{(w_i^k)^2} \left| \sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,k} x_j^0 - \sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,k} \frac{1}{N} \sum_{l=1}^{N} x_l^0 \right|^2 \\ &\leq \Psi_1^k \Psi_2^k \qquad (3.21) \end{aligned}$$

$$\text{with} \qquad \Psi_1^k = \frac{\|x^0\|_2^2}{[\min_i w_i^k]^2} \qquad (3.22)$$

$$\Psi_2^k = \sum_{i=1}^{N}\sum_{j=1}^{N} \left| \left(\mathbf{P}^{1,k}(\mathbf{I}-\mathbf{J})\right)_{i,j} \right|^2 = \left\| \mathbf{P}^{1,k}\mathbf{J}^{\perp} \right\|_{\text{F}}^2. \qquad (3.23)$$

From now, our main contributions will be to understand the asymptotic behavior of both terms Ψ_1^k and Ψ_2^k. In Section 3.4.2, we will prove that there is a constant $C < \infty$ such that the event $\{\Psi_1^k \leq C\}$ occurs infinitely often with probability 1. The term Ψ_2^k represents the projection of the current sensor values on the orthogonal space to the consensus line; the exponential decay of this term will be proven in Section 3.4.3. Finally, we will put together these two results in Section 3.4.4 to derive our core results.

3.4.2 Analysis of $\Psi_1(t)$

This term depends on the inverse of the minimum of the sensors weights (see Eq. (3.22)) and thus can increase quickly if one of the weights goes close to zero. However, the sensors frequently exchange information and hence spread their weight so the probability that a node weight keeps decreasing for a long time should be very small. We will work on the probability that every sensor weight is greater than a positive constant and show that this event occurs infinitely often. This will enable us to prove that there exists $C < \infty$ such that $\mathbb{P}[\{\Psi_1^k \leq C\} \text{ infinitely often}] = 1$. To obtain these results, some preliminary lemmas are needed.

Lemma 3.8. *Under Assumption 3.7, for any $s < k$,*

$$\text{if } \mathbf{P}^{s,k} \geq c\mathbb{1}\mathbb{1}^{\text{T}} \text{ for some } c > 0 \text{ then } w^k \geq cN\mathbb{1}.$$

Proof. First let us remark that as the matrices of $\mathscr{K}$ are column-stochastic from Assumption 3.7 and $w^0 = \mathbb{1}$, we have that w^s is non-negative and $\mathbb{1}^{\mathrm{T}} w^s = N$ for all $s > 0$. Hence, if $\mathbf{P}^{s,k} \geq c\mathbb{1}\mathbb{1}^{\mathrm{T}}$ for some $c > 0$, then $w^k = \mathbf{P}^{s,k} w^s \geq c\mathbb{1}\mathbb{1}^{\mathrm{T}} w^s = cN\mathbb{1}$ (see [3, Chap. 8.1] for details about inequalities for non-negative matrices). □

It is thus interesting to focus on the minimal value of $\mathbf{P}^{s,k}$. To this purpose, we will i) give a lower-bound on the smallest non-null value of $\mathbf{P}^{s,k}$; and then ii) prove that there is a time L such that the probability that $\mathbf{P}^{s,s+L} > 0$ is positive.

Lemma 3.9. *Under Assumption 3.7, for all $1 \leq s < k$ and i, j,*

$$\mathbf{P}^{s,k}_{i,j} = 0 \text{ or } \mathbf{P}^{s,k}_{i,j} \geq (m_{\mathscr{K}})^{k-s+1}$$

where $m_{\mathscr{K}} = \min_{i,j} \left\{ \mathbf{K}_{i,j} : \mathbf{K} \in \mathscr{K}, \mathbf{K}_{i,j} > 0 \right\}$ is the smallest non-null entry of all the matrices of the set $\mathscr{K}$.

Proof. Let us consider the random matrix $\mathbf{P}^{1,k}$ (as the matrix choice is i.i.d., we drop the offset s). We will prove this result by induction. It is trivial to see that every non-null coefficient of $\mathbf{P}^{1,1} = \mathbf{K}_{\xi^1}$ is greater than $m_{\mathscr{K}}$ and as

$$\mathbf{P}^{1,k}_{i,j} = \sum_{l=1}^{N} (\mathbf{K}_{\xi^k})_{i,l} \mathbf{P}^{1,k-1}_{l,j},$$

it is obvious that if $\mathbf{P}^{1,k}_{i,j} > 0$, there is a term in the above sum that is positive (we recall that all the coefficient here are non-negative). This term is the product of a positive coefficient of $\mathbf{K}_{\xi^k}$ and a positive coefficient of $\mathbf{P}^{1,k-1}$. Hence, if all the non-null coefficients of $\mathbf{P}^{1,k-1}$ are greater than $(m_{\mathscr{K}})^k$, then any non-null coefficient of $\mathbf{P}^{1,k}$ is greater than $(m_{\mathscr{K}})^k . m_{\mathscr{K}} = (m_{\mathscr{K}})^{k+1}$. So, by induction, we have that $\forall k > 1$ every non-null coefficient of $\mathbf{P}^{1,k}$ is greater than $(m_{\mathscr{K}})^{k+1}$. □

Now that we have lower bounded the positive coefficients of the product matrices, we have to investigate the instants where these matrices are positive.

Lemma 3.10. *Under Assumption 3.7, there is a finite constant L such that for all $s > 0$,*

$$\mathbb{P}[\mathbf{P}^{s,s+L} > 0] > 0.$$

Proof. As $\{\xi^k\}_{k>0}$ is an i.i.d. sequence and $\mathbb{E}[\mathbf{K}]$ is primitive from Assumption 3.7, there is a finite m such that $\mathbb{E}[\mathbf{K}]^m > 0$ and $\mathbb{E}[\mathbf{K}_{\xi^m} \mathbf{K}_{\xi^{m-1}} \dots \mathbf{K}_{\xi^1}] = \mathbb{E}[\mathbf{K}]^m > 0$ so $\mathbb{P}[(\mathbf{K}_{\xi^m} \mathbf{K}_{\xi^{m-1}} \dots \mathbf{K}_{\xi^1})_{i,j} > 0] > 0$ for any entry (i, j). Furthermore, $\mathbf{P}^{s,k}_{i,j} > 0$ implies that $\mathbf{P}^{s,k'}_{i,j} > 0$ for all $k' \geq k$ because all the matrices of $\mathscr{K}$ have positive diagonal elements (indeed $\mathbf{P}^{s,k'} = \mathbf{P}^{k+1,k'} \mathbf{P}^{s,k} = (\varepsilon \mathbf{I} + (\mathbf{P}^{k+1,k'} - \varepsilon \mathbf{I}))\mathbf{P}^{s,k} = \varepsilon \mathbf{P}^{s,k} + (\mathbf{P}^{k+1,k'} - \varepsilon \mathbf{I})\mathbf{P}^{s,k}$ and as one can find $\varepsilon > 0$ so that the second matrix is non-negative, $\mathbf{P}^{s,k'}_{i,j}$ is positive if $\mathbf{P}^{s,k}_{i,j}$ is). Finally, by taking $L = mN^2$, we get that the probability that

all entries of $\mathbf{P}^{s,s+L}$ are positive is greater than the probability of $\{\mathbf{P}_{1,1}^{s,s+m}>0\}\cap\{\mathbf{P}_{1,2}^{s+m+1,s+2m}>0\}\cap\cdots\cap\{\mathbf{P}_{N,N}^{s+L-m+1,s+L}>0\}$ so

$$\mathbb{P}[\mathbf{P}^{s,s+L}>0]\geq\prod_{i,j=1}^{N}\mathbb{P}[(\mathbf{P}^{s+m[(i-1)+(j-1)N]+1,s+m[i+(j-1)N]})_{i,j}>0]>0$$

which concludes the proof. □

Under Assumption 3.7 and where L be the same constant as in Lemma 3.10, we define $E_n\triangleq\{\mathbf{P}^{nL,(n+1)L}>0\}$ as being the event where the product matrix is positive considering iterations L by L. We also define the following times:

$$\begin{cases}\tau_0=0\\ \tau_n=L\times\min\left\{j:\sum_{\ell=1}^{j}\mathbb{I}_{E_\ell}=n\right\}\end{cases}$$

where $\mathbb{I}_E$ is the indicator function[4] of event E. And,

$$\Delta_n=\tau_n-\tau_{n-1}\qquad n=1,...,\infty.$$

The events $\{E_n\}_{n>0}$ are i.i.d. and $\mathbb{P}[E_1]>0$ according to Lemma 3.10, so Borel-Cantelli lemma tells us that E_n occurs infinitely often. Hence, the inter-arrival times $\{\Delta_n\}_{n>0}$ are i.i.d. and geometrically distributed up to a multiplicative factor L *i.e.* $\mathbb{P}[\Delta_1=jL]=p^{j-1}(1-p)$ for $j\geq1$ and $p\in(0,1)$. Finally, observe that the $\{\tau_n\}_{n>0}$ are all finite and the sequence converges to infinity with probability one. These results along with Lemmas 3.8 and 3.9 enable us to state the final result on $\{\Psi_1^k\}_{k>0}$.

Proposition 3.11. *Under Assumption 3.7, there is a finite constant L such that there exists a sequence of positive i.i.d. geometrically distributed random variables $\{\Delta_n\}_{n>0}$ such that for all $n>0$,*

$$\Psi_1^{\tau_n}\leq\|x^0\|_2^2(m_{\mathscr{K}})^{-2L}$$

where $\tau_n=\sum_{\ell=1}^{n}\Delta_\ell$.

3.4.3 Analysis of $\Psi_2(t)$

This section deals with the exponential decay of $\{\Psi_2^k\}_{k>0}$; these results extend significantly those given in [43] since we consider a more general model for $\mathscr{K}$ and $\{\xi^k\}_{k<0}$. According to Eq. (3.23), we have, for all $k>0$,

$$\Psi_2^k=\|\mathbf{P}^{1,k}\mathbf{J}^{\perp}\|_{\mathrm{F}}^2. \tag{3.24}$$

The technique developed in Section 3.2.2-b (and used in *e.g.* [6]) which is based on the spectral norm can be mimicked. We write

$$\begin{aligned}\mathbb{E}[\Psi_2^{k+1}|\Psi_2^k] &= \mathbb{E}[\mathrm{Trace}\left(\mathbf{J}^{\perp}(\mathbf{P}^{1,k+1})^{\mathrm{T}}\mathbf{P}^{1,k+1}\mathbf{J}^{\perp}\right)|\mathbf{P}^{1,k}]\\ &= \mathrm{Trace}\left(\mathbf{J}^{\perp}(\mathbf{P}^{1,k})^{\mathrm{T}}\mathbf{J}^{\perp}\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]\mathbf{J}^{\perp}\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right) &(3.25)\\ &\leq \rho(\mathbf{J}^{\perp}\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]\mathbf{J}^{\perp})^2\ \Psi_2^k &(3.26)\end{aligned}$$

[4] that is the function equal to 1 if E is true and zero otherwise.

where the first equality comes from the identity $\|\mathbf{X}\|_{\mathrm{F}}^2 = \mathrm{Trace}(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ for any real matrix, the second comes from the linearity of the trace. The final inequality comes from the fact that the trace can be seen as the sum of the eigenvalues and $\mathbf{X}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\mathbf{X} \preccurlyeq \rho(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})\mathbf{X}^{\mathrm{T}}\mathbf{X}$ where '$\preccurlyeq$' denotes an inequality in the semi-definite ordering (see [3, Chap. 7.7]).

Unfortunately, this proof technique does not work in the most general case[5]. As a consequence, this inequality is not tight enough to prove a general convergence result and another recursion has to be found.

3.4.3-a A new and tighter recursion matrix: R

Therefore, as proposed alternatively in [6] (though not essential in [6]) in the context of Random Gossip algorithm, we write Ψ_2^k with respect to a more complicated matrix for which the recursion property is tighter. Indeed, recalling that for any real matrix $\mathbf{X}$,

$$\begin{aligned} \|\mathbf{X}\|_{\mathrm{F}}^2 &= \mathrm{Trace}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right) \\ \text{and} \quad \mathrm{Trace}\left(\mathbf{X}\otimes\mathbf{X}\right) &= \left(\mathrm{Trace}\left(\mathbf{X}\right)\right)^2 \end{aligned}$$

where '$\otimes$' denotes the *Kronecker product*, one can find that

$$\begin{aligned} \|\mathbf{X}\|_{\mathrm{F}}^2 &= \mathrm{Trace}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right) = \sqrt{\mathrm{Trace}\left((\mathbf{X}^{\mathrm{T}}\mathbf{X})\otimes(\mathbf{X}^{\mathrm{T}}\mathbf{X})\right)} \\ &= \sqrt{\mathrm{Trace}\left((\mathbf{X}^{\mathrm{T}}\otimes\mathbf{X}^{\mathrm{T}})(\mathbf{X}\otimes\mathbf{X})\right)} = \sqrt{\mathrm{Trace}\left((\mathbf{X}\otimes\mathbf{X})^{\mathrm{T}}(\mathbf{X}\otimes\mathbf{X})\right)} \\ &= \|\mathbf{X}\otimes\mathbf{X}\|_{\mathrm{F}}. \end{aligned}$$

So, we have that

$$\Psi_2^k = \|\boldsymbol{\Xi}^k\|_{\mathrm{F}}$$

with

$$\boldsymbol{\Xi}^k \triangleq \left(\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right)\otimes\left(\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right). \tag{3.27}$$

As before, we easily see that for a column stochastic matrix $\mathbf{K}$, one has $\mathbf{J}^{\perp}\mathbf{K}\mathbf{J}^{\perp} = (\mathbf{I}-\mathbf{J})\mathbf{K}\mathbf{J}^{\perp} = (\mathbf{K}-\mathbf{J})\mathbf{J}^{\perp} = \mathbf{K}\mathbf{J}^{\perp}$. Hence, using standard properties on the Kronecker product, we have

$$\begin{aligned} \boldsymbol{\Xi}^{k+1} &= \left(\mathbf{K}_{\xi^{k+1}}\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right)\otimes\left(\mathbf{K}_{\xi^{k+1}}\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right) \\ &= \left(\mathbf{K}_{\xi^{k+1}}\mathbf{J}^{\perp}\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right)\otimes\left(\mathbf{K}_{\xi^{k+1}}\mathbf{J}^{\perp}\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right) \\ &= \left(\mathbf{K}_{\xi^{k+1}}\otimes\mathbf{K}_{\xi^{k+1}}\right)\left(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}\right)\boldsymbol{\Xi}^k. \end{aligned} \tag{3.28}$$

By considering the mathematical expectation given the natural filtration of the past events $\mathscr{F}_k = \sigma\left(\xi^1,\ldots,\xi^k\right)$, we obtain

$$\mathbb{E}\left[\boldsymbol{\Xi}^{k+1}|\mathscr{F}_k\right] = \mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]\left(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}\right)\boldsymbol{\Xi}^k$$

[5] Sometimes, this spectral radius of $\mathbf{J}^{\perp}\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]\mathbf{J}^{\perp}$ can be greater than 1; indeed for the *BWGossip* algorithm (introduced later in Section 3.5.1), one can have $\|\|\mathbf{J}^{\perp}\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]\mathbf{J}^{\perp}\|\|_2 > 1$ for some underlying graphs even is the algorithm converges as we will see later.

and so we have

$$\mathbb{E}\left[\mathbf{\Xi}^k\right] = (\mathbf{R})^k \tag{3.29}$$

with

$$\mathbf{R} \triangleq \mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]\left(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}\right). \tag{3.30}$$

We thus have to analyze the behavior of $(\mathbf{R})^k$ as k goes to infinity.

3.4.3-b Relationship between the recursion matrix R and the convergence speed

Now, let us find a simple relationship between $\mathbb{E}[\Psi_2^k]$ and the entries of the matrix $\mathbb{E}[\mathbf{\Xi}^k]$ by considering $\mathbf{Q}^k \triangleq \mathbf{P}^{1,k}\mathbf{J}^{\perp}$. We show that

$$\left(\mathbb{E}[\mathbf{\Xi}^k]\right)_{i+(l-1)N,j+(m-1)N} = \mathbb{E}[\mathbf{Q}_{ij}^k\mathbf{Q}_{lm}^k],\ \forall i,j,l,m\in\{1,\ldots,N\}.$$

According to Eq. (3.24), we have $\mathbb{E}[\Psi_2^k] = \mathbb{E}[\|\mathbf{Q}^k\|_{\mathrm{F}}^2]$ which implies that

$$\mathbb{E}[\Psi_2^k] = \sum_{i,j=1}^{N}\mathbb{E}\left[(\mathbf{Q}_{ij}^k)^2\right] = \sum_{i,j=1}^{N}\left(\mathbb{E}\left[\mathbf{\Xi}^k\right]\right)_{i+(i-1)N,j+(j-1)N}. \tag{3.31}$$

As a consequence, the behavior of the entries of $\mathbb{E}[\mathbf{\Xi}^k]$ drives the behavior of $\mathbb{E}[\Psi_2^k]$.

Using the Jordan normal form of $\mathbf{R}$ (see [3, Chap. 3.1 and 3.2]) and the l_∞ vector norm on matrices $\|\mathbf{X}\|_\infty = N\max_{i,j=1,\ldots,N}|\mathbf{X}_{i,j}|$ (see [3, Chap. 5.7]), we get that there is an invertible matrix $\mathbf{S}$ such that

$$\|(\mathbf{R})^k\|_\infty = \|\mathbf{S}(\mathbf{\Lambda})^k\mathbf{S}^{-1}\|_\infty \le \|\mathbf{S}\|_\infty\|\mathbf{S}^{-1}\|_\infty\|(\mathbf{\Lambda})^k\|_\infty \tag{3.32}$$

where $\mathbf{\Lambda}$ is the Jordan matrix associated with $\mathbf{R}$.

After some computations, we derive that the absolute value of all the entries of $(\mathbf{\Lambda})^k$ is bounded in the following way:

$$\left\|(\mathbf{\Lambda})^k\right\|_\infty \le \max_{0\le j\le J-1}\binom{k}{k-j}\rho(\mathbf{R})^{k-j} \le k^{J-1}\rho(\mathbf{R})^{k-J+1} \tag{3.33}$$

where J is the size of the greatest Jordan block. Note that when $\mathbf{R}$ is diagonalizable, $J=1$, and we get that

$$\left\|(\mathbf{\Lambda})^k\right\|_\infty \le \rho(\mathbf{R})^k \quad \text{(when } \mathbf{R} \text{ is diagonalizable)}. \tag{3.34}$$

Putting together Eqs. (3.29), (3.31), (3.32), (3.33), and remarking that the subspace spanned by $\mathbb{1}\otimes\mathbb{1}$ is in the kernel of $\mathbf{R}$, we get that the size of the greatest Jordan block is smaller than $N-1$, we can derive a fundamental result on the convergence speed of $\mathbb{E}[\Psi_2^k]$.

Lemma 3.12. *We have*

$$\mathbb{E}[\Psi_2^k] = \mathcal{O}\left(k^{N-2}\rho(\mathbf{R})^k\right)$$

where $\mathbf{R}$ *is defined in Eq. (3.30).*

This lemma states that $\mathbb{E}[\Psi_2^k]$ globally decreases exponentially[6] at speed $\rho(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]\,(\mathbf{J}^\perp\otimes\mathbf{J}^\perp)$. We will thus focus on this spectral radius and particularly on the conditions under which it is smaller than 1.

Remark 3.13. *As $\mathbb{E}[\boldsymbol{\Xi}^k]=(\mathbf{R})^k$, we have from [3, Thm. 5.6.12] that*

$$\mathbb{E}\left[\boldsymbol{\Xi}^k\right]\underset{k\to\infty}{\longrightarrow}0 \;\Leftrightarrow\; \rho(\mathbf{R})<1.$$

Furthermore, we get from Eq. (3.31) that $\mathbb{E}\left[\boldsymbol{\Xi}^k\right]\longrightarrow_{k\to\infty}0$ if and only if $\mathbb{E}[\Psi_2^k]\longrightarrow_{k\to\infty}0$. The spectral radius of $\mathbf{R}$ thus plays a central role in the convergence and convergence speed of $\{\Psi_2^k\}_{k>0}$.

3.4.3-c Analysis of the spectral radius of R

The next step of our analysis is to prove that the spectral radius $\rho(\mathbf{R})$ is strictly less than 1 when Assumption 3.7 hold. For this, we will prove that $\mathbb{E}\left[\boldsymbol{\Xi}^k\right]$ converges to zero using another matrix recursion and use Remark 3.13 to conclude on $\rho(\mathbf{R})$.

Actually, one can find a simple linear recursion on $\boldsymbol{\Xi}^k(t)$ (different from the one exhibited in Eq. (3.28)) as

$$\boldsymbol{\Xi}^{k+1}=\left(\mathbf{K}_{\xi^{k+1}}\otimes\mathbf{K}_{\xi^{k+1}}\right)\,\boldsymbol{\Xi}^k$$

thus by taking the mathematical expectation given the past we obtain

$$\mathbb{E}\left[\boldsymbol{\Xi}^{k+1}|\mathscr{F}_k\right]=\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]\,\boldsymbol{\Xi}^k.$$

Remarking that $(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}\boldsymbol{\Xi}^k=0$, we have for any vector ν of $\mathbb{R}^{N^2}$,

$$\mathbb{E}\left[\boldsymbol{\Xi}^{k+1}|\mathscr{F}_k\right]=\left(\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]-\nu(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}\right)\,\boldsymbol{\Xi}^k$$

and then,

$$\mathbb{E}\left[\boldsymbol{\Xi}^k\right]=(\mathbf{T}_\nu)^k\,\boldsymbol{\Xi}^0 \tag{3.35}$$

with $\mathbf{T}_\nu=\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]-\nu(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}$.

By considering Eq. (3.35), it is straightforward that $\mathbb{E}\left[\boldsymbol{\Xi}^k\right]$ converges to zero as k goes to infinity if there is a vector ν such that $\rho(\mathbf{T}_\nu)<1$. However, this condition is only sufficient whereas the one derived from Eq. (3.29) is a necessary and sufficient condition.

The following lemma is very important in our proof as it ensures that, under Assumption 3.7, there is a vector ν such that $\rho\left(\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]-\nu(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}\right)<1$.

Lemma 3.14. *Under Assumption 3.7, there is a vector ν such that*

$$\rho\left(\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]-\nu(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}\right)<1$$

and thus $\mathbb{E}\left[\boldsymbol{\Xi}^k\right]\longrightarrow_{k\to\infty}0$.

[6] as in log scale $\log(k)+k\log(a)\sim_{k\to\infty}k\log(a)$. Furthermore, we could have directly derived that $\forall\varepsilon>0$, $\mathbb{E}[\Psi_2^k]=\mathscr{O}\left((\rho(\mathbf{R})+\varepsilon)^k\right)$ from Eqs. (3.29), (3.31) and [3, Corollary 5.6.13] but this introduces an epsilon that might make the following results less clear.

Proof. Assumption 3.7 implies that:

i) $\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]$ is a non-negative column-stochastic matrix so $\rho(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}])=1$ according to [3, Lemma 8.1.21];

ii) according to Lemma 3.10 there is a finite L such that $\mathbb{P}[\mathbf{P}^{1,L}>0]=\mathbb{P}[\mathbf{P}^{1,L}\otimes\mathbf{P}^{1,L}>0]>0$ so $(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}])^L>0$ which imply that $\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]$ is a primitive matrix.

These two properties imply that 1 is the only eigenvalue of maximal modulus of $\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]$ and it is associated with the left eigenvector $\mathbb{1}\otimes\mathbb{1}$. This means that by taking $v=v_1$ the right eigenvector corresponding to eigenvalue 1 such that $(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}v_1=1$, we get that the spectrum of $\mathbf{T}_{v_1}$ is the same as the one of $\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]$ without the maximal eigenvalue equal to 1. As a consequence, the modulus of the eigenvalues of $\mathbf{T}_{v_1}$ is strictly less than 1, *i.e.* $\rho(\mathbf{T}_{v_1})<1$. □

Finally, Remark 3.13 enables us to conclude on the spectral radius of $\mathbf{R}$.

Corollary 3.15. *Under Assumption 3.7,*

$$\rho(\mathbf{R})<1$$

with $\mathbf{R}$ *defined in Eq. (3.30).*

Combining Lemma 3.12 and Corollary 3.15 concludes our analysis of $\{\Psi_2^k\}_{k>0}$.

Proposition 3.16. *Under Assumption 3.7,*

$$\mathbb{E}[\Psi_2^k]=\mathcal{O}\left(k^{N-2}\mathrm{e}^{-\omega k}\right)$$

with $\omega=-\log(\rho(\mathbf{R}))>0$ *and* $\mathbf{R}$ *defined in Eq. (3.30).*

3.4.4 Final results

Thanks to the various intermediate Lemmas and Propositions provided above, we are now able to state our general theorems for the convergence and convergence speed of Sum-Weight-based averaging algorithms.

3.4.4-a Result on the convergence

First, let us prove an interesting result: the estimates $\{x^k\}_{k>0}$ get closer from each other in the sense of the l_∞ norm and thus the error $\{x^k-x_{\mathrm{ave}}\mathbb{1}\}_{k>0}$ is non-increasing in terms of l_∞ norm.

Lemma 3.17. *Under Assumption 3.7,* $\|x^k-x_{ave}\mathbb{1}\|_\infty=\max_i|x_i^k-x_{ave}|$ *forms a non-increasing sequence with respect to* k.

Proof. One can remark that, at time $k+1$, for all i,

$$\begin{aligned}x_i^{k+1} &= \frac{\sum_{j=1}^N \mathbf{K}_{i,j}s_j^k}{\sum_{j=1}^N \mathbf{K}_{i,j}w_j^k}=\frac{\sum_{j=1}^N \mathbf{K}_{i,j}w_j^k x_j^k}{\sum_{j=1}^N \mathbf{K}_{i,j}w_j^k}\\ &= \sum_{j=1}^N\left(\frac{\mathbf{K}_{i,j}w_j^k}{\sum_{l=1}^N \mathbf{K}_{i,l}w_l^k}\right)x_j^k\end{aligned}$$

where $\mathbf{K}$ corresponds to any matrix in $\mathscr{K}$. Hence, x_i^{k+1} is a center of mass of $\{x_j^k\}_{j=1,\dots,N}$. Therefore, using the fact that $x_{\text{ave}}\mathbb{1}$ verifies the same above inequality as x^{k+1}, we have that $\forall i = 1,\dots,N$,

$$\begin{aligned} |x_i^{k+1} - x_{\text{ave}}| &\leq \sum_{j=1}^{N} \left(\frac{\mathbf{K}_{i,j} w_j^k}{\sum_{l=1}^{N} \mathbf{K}_{i,l} w_l^k} \right) |x_j^k - x_{\text{ave}}| \\ &\leq \max_j |x_j^k - x_{\text{ave}}| \end{aligned}$$

which implies that $\|x^{k+1} - x_{\text{ave}}\mathbb{1}\|_\infty \leq \|x^k - x_{\text{ave}}\mathbb{1}\|_\infty$. □

The above proof also tells us that *Sum-Weight* based algorithms can be seen as a standard averaging algorithm on x with time-varying weighted averages at each agent. Now, let us use the results derived in Sections 3.4.2 and 3.4.3 to state our convergence theorem under necessary and sufficient conditions.

Theorem 3.18. *Under Assumption 3.7,*

$$\{x^k\}_{k>0} \text{ is bounded and converges to the average consensus } x_{ave}\mathbb{1} \text{ almost surely.}$$

Furthermore, if Assumption 3.7c does not hold, $\{x^k\}_{k>0}$ does not converge to the average consensus with probability one for some values of x^0.

Proof. We divide the proof into three steps: i) the boundedness of the sequence; ii) the almost sure convergence; and iii) the fact that Assumption 3.7c is necessary.

⋄ **boundedness:** As $\{\|x^k - x_{\text{ave}}\mathbb{1}\|_\infty\}_{k>0}$ is non-increasing from Lemma 3.17, it is obvious that for any $k > 0$, $\|x^k - x_{\text{ave}}\mathbb{1}\|_\infty \leq \|x^0 - x_{\text{ave}}\mathbb{1}\|_\infty$ thus $\{x^k\}_{k>0}$ is bounded.

⋄ **almost sure convergence:** Let us assume that Assumption 3.7 holds. Using Markov's inequality along with Proposition 3.16, there is a finite constant C such that for any $\varepsilon > 0$,

$$\begin{aligned} \sum_{k>0} \mathbb{P}[\Psi_2^k > \varepsilon] &\leq \frac{1}{\varepsilon} \sum_{k>0} \mathbb{E}[\Psi_2^k] \\ &\leq \frac{1}{\varepsilon} C \sum_{k>0} k^{N-2} \mathrm{e}^{-\omega k} < \infty. \end{aligned}$$

Thus, Borel-Cantelli's lemma leads to the almost sure convergence of $\{\Psi_2^k\}_{k>0}$ to zero. In addition, as the random variables $\{\tau_n\}_{n>0}$ provided in the statement of Proposition 3.11 converge to infinity with probability one, $\Psi_2^{\tau_n} \to_{n\to\infty} 0$ almost surely. Since $\Psi_1^{\tau_n}$ is (upper) bounded for any $n > 0$,

$$\Psi_1^{\tau_n} \Psi_2^{\tau_n} \underset{n\to\infty}{\longrightarrow} 0 \quad \text{almost surely.}$$

According to Lemma 3.17, $\|x^k - x_{\text{ave}}\mathbb{1}\|_\infty$ is a non-increasing nonnegative sequence verifying $\|x^k - x_{\text{ave}}\mathbb{1}\|_\infty^2 \leq \|x^k - x_{\text{ave}}\mathbb{1}\|_2^2 \leq \Psi_1^k \Psi_2^k$ and there is a converging subsequence with limit 0 with probability one (following the $\{\tau_n\}_{n>0}$). As a consequence, the sequence $\{\|x^k - x_{\text{ave}}\mathbb{1}\|_\infty\}_{k>0}$ converges almost surely to the 0 which imply the almost sure convergence of $\{x^k\}_{k>0}$ to the average consensus.

⋄ **Assumption 3.7c is necessary:** Let us consider that Assumption 3.7c does not hold. Recalling Lemma 3.10 and its proof, it is easy to see that if Assumption 3.7c is not true, then $\nexists L$ such that $\mathbb{P}[\mathbf{P}^{1,L} > 0] > 0$ so $\mathbb{P}[\mathbf{P}^{1,k} > 0] = 0$ for any k with probability 1. Let i, j be such that $\mathbf{P}^{1,k}_{i,j} = 0$ for any $k > 0$ (As from Assumption 3.7a the matrices of $\mathscr{K}$ have positive diagonals if $\mathbf{P}^{1,k}_{i,j} = 0$ then $\mathbf{P}^{1,s}_{i,j} = 0$ for any $s \leq k$), the i-th component of $\{x^k_i\}_{k>0}$ is independent of x^0_j hence $\{x^k_i\}_{k>0}$ cannot converge to x_{ave} for all x^0. □

Remark 3.19. *One can note that the previous theorem implies that $\{x^k\}_{k>0}$ converges to the average consensus $x_{ave}\mathbb{1}$ in L^p for any $p \in \mathbb{N}$ by an immediate consequence of the dominated convergence theorem. As a particular case, the Mean Squared Error (MSE) converges to* 0.

3.4.4-b Result on the convergence speed

The next result on the convergence speed corresponds to the main challenge and novelty of our contribution to averaging algorithms. For this theorem we introduce the following notation: given two sequences of random variables $\{X^k\}_{k>0}$ and $\{Y^k\}_{k>0}$, we will say that $X^k = o_{\text{a.s.}}(Y^k)$ if $X^k/Y^k \to_{k\to\infty} 0$ almost surely.

Theorem 3.20. *Under Assumptions 3.7, the Squared Error (SE) is upper-bounded by a non-increasing sequence converging to* 0*. Furthermore, it is also upper-bounded by an exponentially decreasing function as follows*

$$\text{SE}^{\tau_n} = \|x^{\tau_n} - x_{ave}\mathbb{1}\|_2^2 = o_{a.s.}\left(\tau_n^N \mathrm{e}^{-\omega\tau_n}\right)$$

with $\omega = -\log\left(\rho\left(\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right]\left(\mathbf{J}^\perp\otimes\mathbf{J}^\perp\right)\right)\right) > 0$ and $\tau_n = \sum_{\ell=1}^n \Delta_\ell$ as defined in Proposition 3.11.

This result tells us that the squared error will vanish exponentially over regularly spaced stopping times (the difference between them is an i.i.d. process) and we derived a lower bound for this speed. The particular behavior of the weights variables in this very general setting does not enable us to provide a more precise result about the squared error (we can just say that the elementwise maximum does not increase in interval $[\tau_n, \tau_{n+1}]$ for any n – see Theorem 3.18 –); however for some particular algorithms (e.g. standard gossip ones) this derivation is possible (see Section 3.6 for more details). We will illustrate the tightness of the exponential decrease constant ω via numerical results in Section 3.5.

Proof. To prove this result we will once again use the decomposition of the squared error introduced in Eq. (3.21). We know from Proposition 3.16 that $\mathbb{E}[k^{-N}\mathrm{e}^{\omega k}\Psi_2^k] = \mathscr{O}(1/k^2)$. By Markov's inequality and Borel-Cantelli's lemma,

$$k^{-N}\mathrm{e}^{\omega k}\Psi_2^k \underset{k\to\infty}{\longrightarrow} 0 \quad \text{almost surely.}$$

Composing with the $\{\tau_n\}_{n>0}$, we get

$$\tau_n^{-N}\mathrm{e}^{\omega\tau_n}\Psi_2^{\tau_n} \underset{n\to\infty}{\longrightarrow} 0 \quad \text{almost surely.}$$

Since there is a finite constant C such that $\forall n > 0, \Psi_1^{\tau_n} \leq C$, we get the claimed result. □

3.5 Proposed algorithm and extensions

We have proven the convergence and given a bound on the convergence speed of Sum-Weight-like averaging algorithms under mild assumptions, it is interesting to see how the Sum-Weight formalism allows us to derive faster averaging algorithms. In Subsection 3.5.1, we propose a new Sum-Weight algorithm using the broadcast nature of the wireless channel which converges and offers remarkable performance. This algorithm is hereafter called *BWGossip*. In Subsection 3.5.3, we introduce a distributed management of the nodes clocks which can improve averaging algorithms. Finally, Subsections 3.5.4 and 3.5.5 provide an extension of this work to distributed sum computation and the case of i.i.d. failures in the communication graph.

3.5.1 BWGossip algorithm

Remarking i) that the broadcast nature of the wireless channel was often not taken into account in the distributed estimation algorithms (apart in [7] but this algorithm does not converge to the average) and ii) that information propagation is much faster while broadcasting compared to pairwise exchanges (see Chapter 2), we derive an algorithm taking into account the broadcast nature of the wireless channel. At each global clock tick, it simply consists in uniformly choosing a sensor that broadcasts its pair of values in an appropriate way (to ensure column-stochasticity); then, the receiving sensors add their received pair of values to their current one. A more algorithmic formulation is presented below.

As mentioned in Section 1.1.2, the agents activate through an i.i.d. process and for any time $k > 0$ and any sensor $i \in V$, $\mathbb{P}[i \text{ activates at global time } k] = 1/N$.

BWGossip

At each clock tick k, let i be the activating node:

► i broadcasts a scaled version of its pair of values to all its neighbors $\left(\frac{s_i(t)}{d_i+1}; \frac{w_i(t)}{d_i+1}\right)$

► Every neighbor $j \in \mathcal{N}_i$ update:

$$\begin{cases} s_j(t+1) = s_j(t) + \frac{s_i(t)}{d_i+1} \\ w_j(t+1) = w_j(t) + \frac{w_i(t)}{d_i+1} \end{cases}$$

► The sensor i updates:

$$\begin{cases} s_i(t+1) = \frac{s_i(t)}{d_i+1} \\ w_i(t+1) = \frac{w_i(t)}{d_i+1} \end{cases}$$

Using a matrix formalism, the iteration at step k is

$$\begin{cases} s^{k+1} = \mathbf{K}_i s^k \\ w^{k+1} = \mathbf{K}_i w^k \end{cases}$$

with

$$\mathbf{K}_i \triangleq \begin{bmatrix} 1 & & & 1/(d_i+1) & & & \\ & 1 & & 1/(d_i+1) & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & 1/(d_i+1) & & & \\ & & & 1/(d_i+1) & 1 & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \\ & & & 1/(d_i+1) & & & & & 1 \end{bmatrix}.$$

This form matches our formalism with $\mathscr{K} = \{\mathbf{K}_i\}_{i\in V}$ and $\xi^k = i$ with probability $1/N$. Obviously, all matrices of $\mathscr{K}$ are column-stochastic but not row-stochastic and they have positive diagonal elements. Again, $\mathrm{Supp}(\mathbb{E}[\mathbf{K}]) = (\mathbf{I}+\mathbf{A})$ with $\mathbf{A}$ the adjacency matrix of the underlying graph and Proposition 1.2 tells us that $\mathbb{E}[\mathbf{K}]$ is primitive as soon as the underlying graph is connected.

This implies that Assumption 3.7 is verified and hence that the BWGossip converges to the average consensus almost surely by Theorem 3.18 and Theorem 3.20 gives us an insight about the decrease speed of the squared error.

3.5.2 Performance of the BWGossip

In order to investigate the performance of distributed averaging algorithms over WSN Wireless Sensor Networks, we plot the MSE obtained by Monte-Carlo simulations versus the number of iterations. The underlying graph is modeled by 100-nodes connected RGGs with radius 2 (see Section 2.5.1 for details about these graphs).

In Fig. 3.1, we compare different average gossip algorithms: i) the *Random Gossip* [11] which is the reference algorithm in the literature (see Section 3.2.2-b for details); ii) the *Broadcast Gossip* [7] which uses the broadcasting abilities of the wireless channel but does not converge to the average (see Section 3.2.3 for details); iii) the algorithm introduced by Franceschelli in [51] which uses a bivariate scheme similar to Sum-Weight and seems to converge (no convergence proof is provided in the paper); and iv) the proposed *BWGossip* algorithm.

We remark that the *BWGossip* algorithm outperforms the existing algorithms without adding routing or any other kind of complexity. Furthermore, the slope is linear in log scale, it is thus interesting to compare this slope to the one derived in Theorem 3.20.

In Fig. 3.2, we display the empirical convergence slope obtained by linear regression on the logarithm of the empirical mean squared error and the associated lower-bound ω derived in Theorem 3.20 for the *BWGossip* algorithm versus the number of sensors N. Different Random Geometric Graphs with same radii $r_0 = 2$ have been considered. We observe that the slope of our proposed bound is tight. Note that as we proposed an upper bound for a exponentially

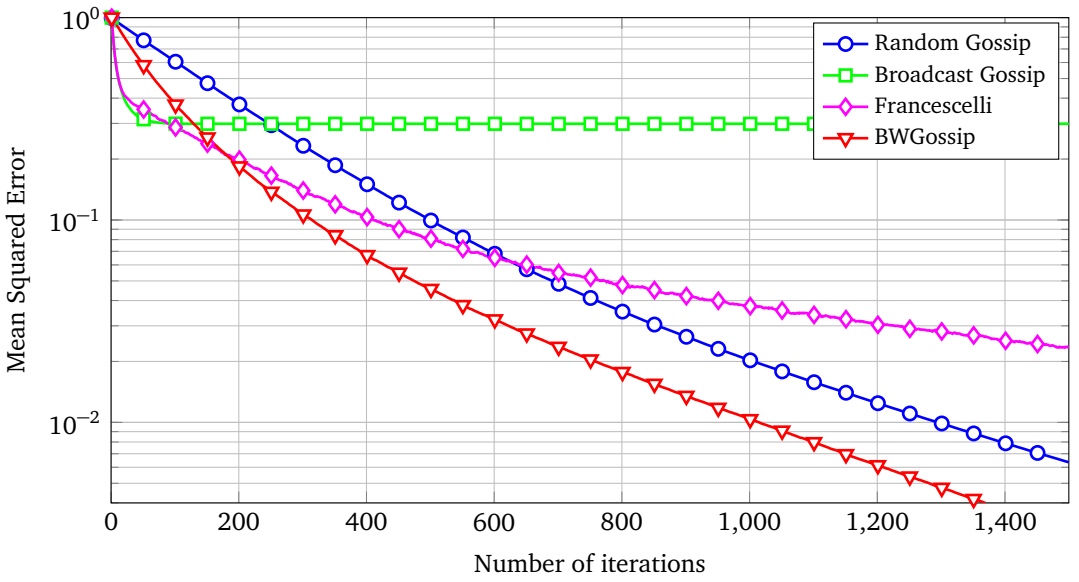


Figure 3.1: Mean squared error of the *BWGossip* and other algorithms of the literature versus time.

decreasing quantity, the associated slope in log scale must be a lower bound of the actual slope as in the displayed figures.

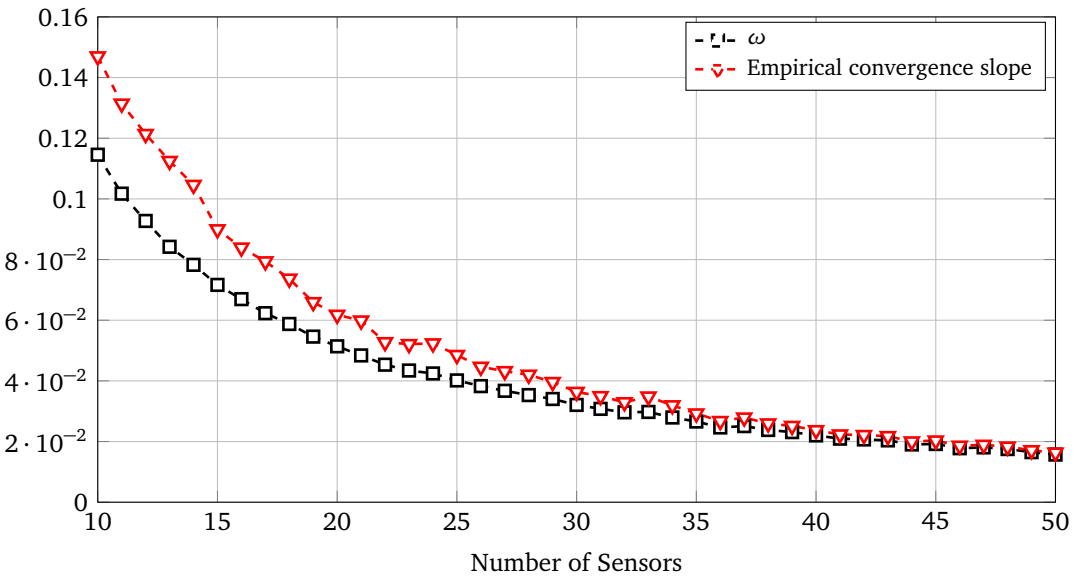


Figure 3.2: Empirical convergence slope of the *BWGossip* and associated lower bound ω.

3.5.3 Adaptation to smart clock management

So far, the choice of the awaking sensor was done uniformly, *i.e.*all the coefficients of the Poisson clocks were identical in our model (see Section 1.1.2). This way, all sensors were

waking up uniformly and independently from their past actions. Intuitively, it would be more logical for a sensor to *talk* less if it has been very active during the past iterations.

Another advantage of the Sum-Weight algorithms is the knowledge of how much a sensor talks compared to the others, which is a useful information. Actually, each sensor knows whether it talks frequently or not (without additional cost) through its own weight value because when a sensor talks, its weight decreases and conversely when it receives information, its weight increases. Therefore, our idea is to control the Poisson coefficient of each sensor with respect to its weight.

We thus propose to consider the following rule for each Poisson coefficient

$$\forall i \in V, \quad \lambda_i^k = \alpha + (1-\alpha) w_i^k \tag{3.36}$$

where $\alpha \in (0,1)$ is a tuning coefficient.

Notice that the global clock remains unchanged since $\forall k > 0$, $\sum_{i=1}^{N} \lambda_i^k = N$. Keeping the global message exchange rate unchanged, the clock rates of each sensor are improved. The complexity of the algorithm is the same because the sensor whose weight changes has just to launch a Poisson clock.

We see in Fig. 3.3 where we plot the empirical mean squared error for the *BWGossip* algorithm versus time with different clock tuning coefficients that even if the convergence and the convergence speed with clock improvement have not been formally established, it seems the exponential convergence to the average consensus still holds, and the convergence can be quicker if α is well chosen. Compared to the algorithm without clock management ($\alpha = 1$), the convergence is much faster at the beginning with $\alpha = 0$ but the asymptotic rate is lower; with $\alpha = 0.5$, the performance is better than the standard BWGossip for every time.

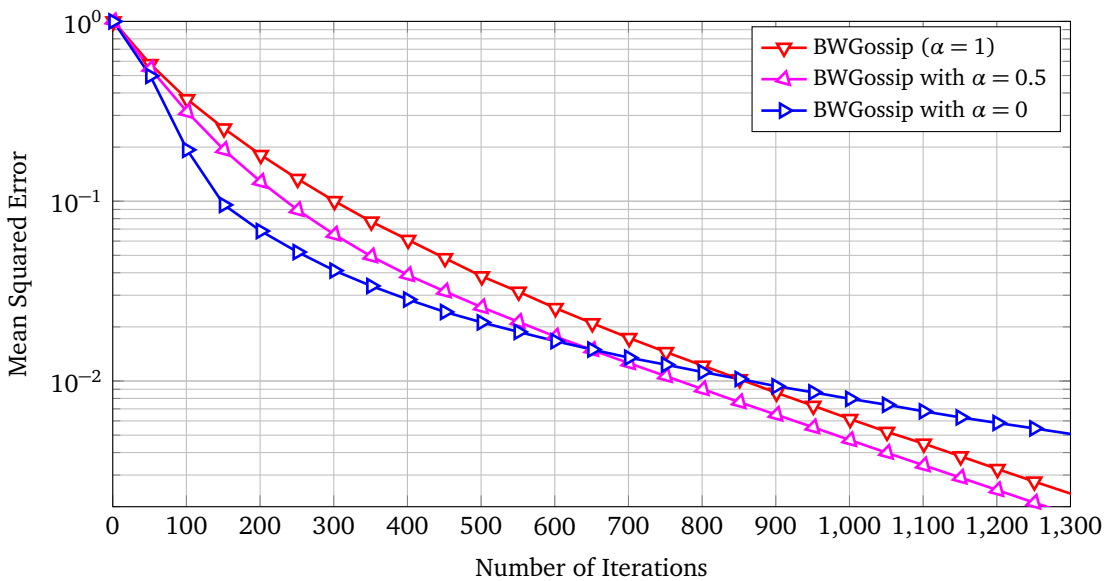


Figure 3.3: Mean squared error of the *BWGossip* with clock management for different values of α.

3.5.4 Distributed estimation of the sum

In some cases, distributively computing the sum of the initial values is very interesting. For example, in the case of signal detection, the Log Likelihood Ratio (LLR) of a set of sensors is separable into the sum of the LLRs of the sensors. Hence, in order to perform a signal detection test based on the information of the whole network (using a Generalized LLR Test for instance), every sensor needs to estimate the sum of the LLRs computed by the sensors (see Section 3.7).

An estimate of the sum can be trivially obtained by multiplying the average estimate by the number of sensors but this information may not be available at any sensor. Another interest of the Sum-Weight scheme is that the initialization of the weights of the sensors enables us to compute different functions related to the average. Intuitively, as the sum of the sequences $\{s^k\}_{k>0}$ and $\{w^k\}_{k>0}$ is conserved through time and the convergence of their ratio to a consensus is guaranteed by Assumption 3.7. Actually as seen in Section 3.3, $\{x^k\}_{k>0}$ converges to a consensus over $\sum_i s_i^0 / \sum_i w_i^0$; which is obviously equal to $x_{\text{ave}} = 1/N \sum_i x_i^0(0)$ with the initialisation of Eq. (3.17).

Now, if a sensor wants to trigger a estimation of the sum through the network, it simply sets its weight to 1 and sends a starting signal to the other nodes which set their weights to 0. Mathematically, we then have the following initialization after sensor i triggers the algorithm

$$\begin{cases} s^0 = x^0 \\ w^0 = e_i \end{cases}$$

where e_i is the i-th canonical vector. In this setting, all Sum-Weight like algorithms converge exponentially to the sum of the initial value as our theorems hold with only minor modifications in the proofs.

Using this initialization, during the first iterations of the algorithm some of the sensors weights are null and hence their estimate is undefined. This can be solved by setting $x_i^k = s_i^k$ while $w_i^k = 0$. The time for the algorithm to be *put on tracks* is equal to the time for the information of the initializing sensor to be spread to everyone and thus has been analyzed in Chapter 2. More precisely, using Random Gossip, this time is equal to the convergence time of Random Pairwise Max (see Section 2.4.2); and using BWGossip, it is equal to the convergence time of Random Broadcast Max (see Section 2.4.3).

3.5.5 Convergence with i.i.d. failures in the communication graph

For some reasons as objects in the light of sight creating a fading in the wireless channel, links in the communication graph can disappear temporarily. Looking at Assumption 3.7, we see that only the expectation of the matrices has to support a strongly connected graph. Let us assume in this section that the underlying graph represented by its adjacency matrix $\mathbf{A}$ can suffer from i.i.d. link failures so that each link e of the graph can fail with probability p_e. Whenever $p_e < 1$, the graph is still connected in mean ($\mathbb{E}[\mathbf{A}]$ is primitive) and thus our conditions are still verified and our analysis holds.

Let us we inspect the influence of link failures in the underlying communication graph on the *BWGossip* algorithm. We consider a connected 10-sensors network onto which i.i.d. link failure events appear with probability p_e (taken identical for each edge e).

In Fig. 3.4, we plot the empirical MSE of the *BWGossip* versus time for different values of the edge failure probability p_e. As expected, we observe that the higher p_e the slower the convergence but the MSE still exponentially decreases.

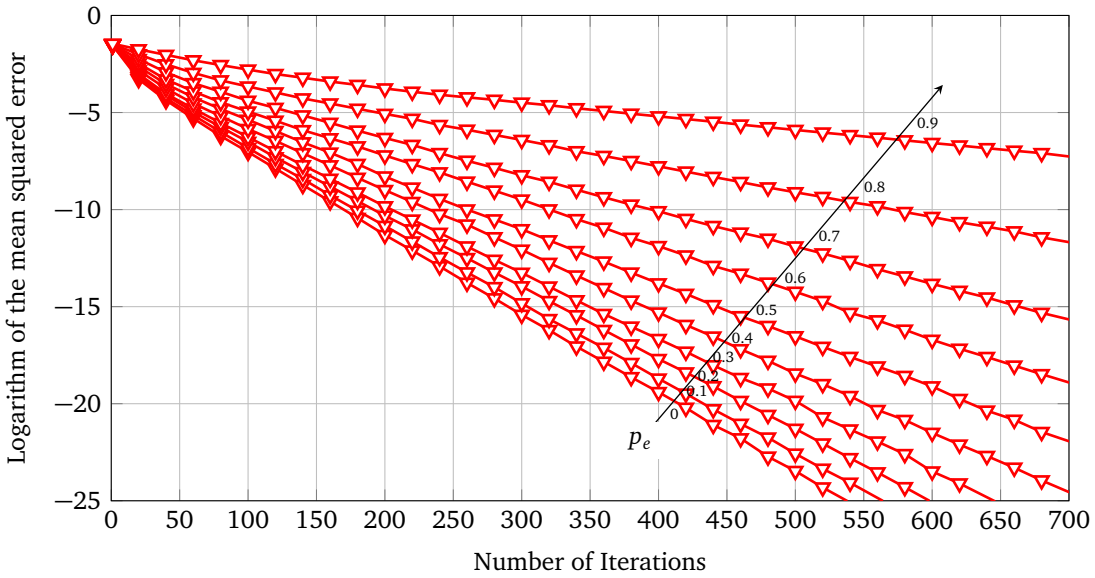


Figure 3.4: Mean squared error of the *BWGossip* for different values of the edge failure probability p_e.

Then, in Fig. 3.5, we plot the empirical convergence slope and the associated bound ω for different link failure probabilities. Here, ω is computed according to a modified matrix set taking into account the link failures through different update matrices. We remark a very good fitting between our lower bound and the simulated results. Consequently, computing ω on the matrix set including the link failures enables us to predict very well the convergence speed in this context.

3.6 Comparison with existing works

In this section, we will show that our results extend the works done previously in the literature. In Subsection 3.6.1 and 3.6.2, we compare our results with existing papers dealing with the design and the analysis of the Sum-Weight like algorithms. In Subsection 3.6.3, we will observe that our results can even be applied to the traditional framework of single-variate gossip algorithms.

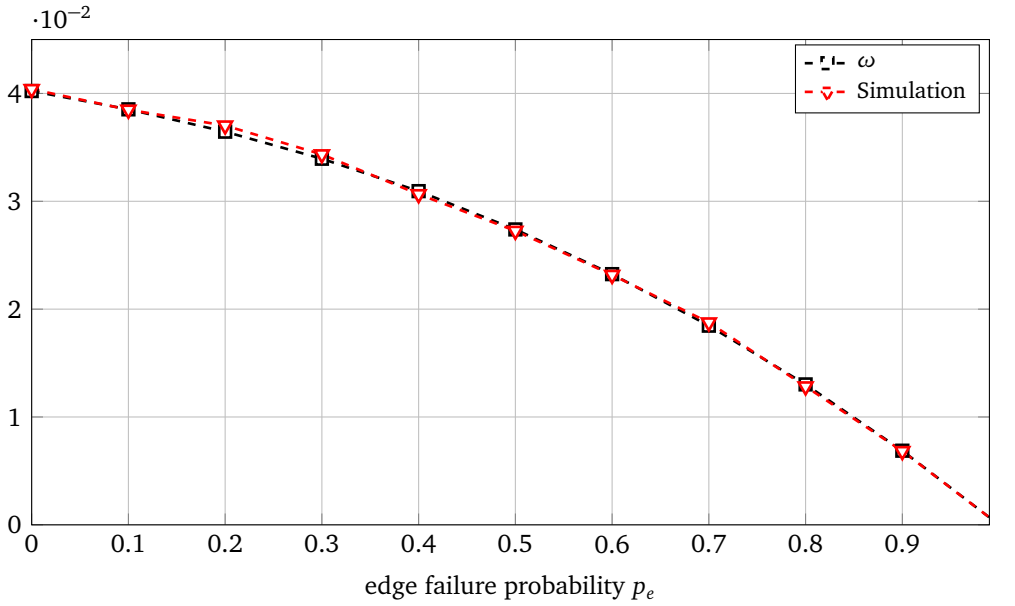


Figure 3.5: Empirical convergence slope and associated bound ω for different values of the edge failure probability p_e.

3.6.1 Comparison with Kempe's algorithm

In Kempe's work [43], the setup is quite different since the sensors update synchronously, that is, at each time k, all the sensors send and update their values. Another important difference lies in the fact that the communication graph is assumed to be complete and to offer self-loops, *i.e.*each sensor can communicate with any other one, including itself. The algorithm introduced in [43] is described below.

Push-Sum Algorithm

At each time k, every sensor i activates:

► i chooses a neighbor (or itself) j_i^{k+1} uniformly in $\mathcal{N}_i \cup \{i\}$ and sends it its pair of values.

► Let $\mathscr{R}_i^{k+1}$ be the set of sensors that sent information to i, i updates:

$$\begin{cases} s_i^{k+1} = \frac{s_i^k}{2} + \sum_{r \in \mathscr{R}_i^{k+1}} \frac{s_r^k}{2} \\ w_i^{k+1} = \frac{w_i^k}{2} + \sum_{r \in \mathscr{R}_i^{k+1}} \frac{w_r^k}{2} \end{cases}$$

Using a matric formalism, the iteration at step k is

$$\begin{cases} s^{k+1} = \mathbf{K}_{j_1^{k+1},\dots,j_N^{k+1}} s^k \\ w^{k+1} = \mathbf{K}_{j_1^{k+1},\dots,j_N^{k+1}} w^k \end{cases}$$

Consequently, at time k, the update matrix takes the following form

$$\mathbf{K}_{j_1^{k+1},\dots,j_N^{k+1}} = \frac{1}{2}\mathbf{I} + \frac{1}{2}\sum_{i=1}^{N} e_{j_i^{k+1}} e_i^{\mathrm{T}}. \tag{3.37}$$

Notice that the first term of the right hand side corresponds to the information kept by the sensor, while the second term corresponds to the information sent to the chosen sensor. Moreover, as each sensor selects uniformly its neighbor[7] (including itself), we immediately obtain that

$$\mathbb{E}[\mathbf{K}] \;=\; \frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{J}.$$

It is then easy to check that i) the update matrices are column-stochastic[8] with a strictly positive diagonal; and ii) that $\mathbb{E}[\mathbf{K}] > 0$, thus it is a primitive matrix. This proves that the Kempe's algorithm satisfies the Assumptions 3.7 and so that it converges almost surely to the average consensus (which was also proven in [43]).

Let us now focus on the convergence speed of the Kempe's algorithm. We remind that the convergence speed is driven by $\{\Psi_2^k\}_{k>0}$ (denoted by $\{\Phi_t\}_{t>0}$ in [43]). As this algorithm is synchronous and only applies on a complete communication graph, it is simple to obtain a recursion between $\mathbb{E}[\Psi_2^{k+1}|\Psi_2^k]$ and Ψ_2^k using the approach of Section 3.2.2-b, and more precisely Eq. (3.25). Indeed, one has

$$\mathbb{E}[(\mathbf{K}\mathbf{J}^{\perp})^{\mathrm{T}}(\mathbf{K}\mathbf{J}^{\perp})] = \left(\frac{1}{2} - \frac{1}{4N}\right)\mathbf{J}^{\perp}, \tag{3.38}$$

(see Appendix B.1 for details). Then, we have that

$$\mathbb{E}[\Psi_2^{k+1}|\Psi_2^k] = \mathrm{Trace}\left(\mathbf{J}^{\perp}(\mathbf{P}^{1,k})^{\mathrm{T}}\mathbb{E}[(\mathbf{K}\mathbf{J}^{\perp})^{\mathrm{T}}(\mathbf{K}\mathbf{J}^{\perp})]\mathbf{P}^{1,k}\mathbf{J}^{\perp}\right) = \left(\frac{1}{2} - \frac{1}{4N}\right)\Psi_2^k. \tag{3.39}$$

Moreover, from Eq. (3.38), we get that $\rho((\mathbf{K}\mathbf{J}^{\perp})^{\mathrm{T}}(\mathbf{K}\mathbf{J}^{\perp})) = 1/2 - 1/(4N) < 1$ and thus the inequality of Eq. (3.26) is replaced with an equality. Therefore, the true convergence speed is provided by $\rho(\mathbb{E}[(\mathbf{K}\mathbf{J}^{\perp})^{\mathrm{T}}(\mathbf{K}\mathbf{J}^{\perp})])$. Comparing this convergence speed[9] (obtained very easily in [43]) with the convergence speed bounds obtained previously is of great interest and will be done below.

First of all, we remind that in the general case treated in this chapter, it is impossible to find a recursion similar to Eq. (3.39) which justifies our alternative approach. Secondly, following the general alternative approach developed previously, the matrix of interest is $\mathbf{R} = \mathbb{E}[\mathbf{K}\otimes\mathbf{K}](\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$ (see Proposition 3.16). After some computations (a detailed proof is available in Appendix B.2), we have that

$$\mathbf{R} = \frac{1}{4}\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp} + \frac{N-1}{4N}vv^{\mathrm{T}} \tag{3.40}$$

[7] as the graph is complete, this means, choosing one node uniformly in the graph.

[8] In Kempe's article [43], the update matrix corresponds to the transpose of the matrix presented here, it is thus row-stochastic in the paper.

[9] Note that there is a typo in Lemma 2.3 of [43]. Indeed, the coefficient is $(1/2-1/(2N))$ in [43] instead of $(1/2-1/(4N))$.

with $v = (1/\sqrt{N-1})(u-(1/N)\mathbb{1}\otimes\mathbb{1})$ and $u = \sum_{i=1}^{N} e_i \otimes e_i$.

Consequently, $\mathbf{R}$ is a linear combination of two following orthogonal projections:

- the first projection, generated by $\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}$, is of rank N^2-2N+1,
- the second projection, generated by vv^{T}, is of rank 1.

As $\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}$ and vv^{T} are orthogonal projections, the vector space $\mathbb{R}^{N^2}$ (on which the matrix $\mathbf{R}$ is operating) can be decomposed into a direct sum of four subspaces:

- $\mathscr{S}_0 = \mathscr{I}m(vv^{\mathrm{T}}) \cap \mathscr{K}er(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$
- $\mathscr{S}_1 = \mathscr{I}m(vv^{\mathrm{T}}) \cap \mathscr{I}m(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$
- $\mathscr{S}_2 = \mathscr{K}er(vv^{\mathrm{T}}) \cap \mathscr{I}m(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$
- $\mathscr{S}_3 = \mathscr{K}er(vv^{\mathrm{T}}) \cap \mathscr{K}er(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$

As $\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}v = v$ (see Appendix B.2), we have $\mathscr{S}_0 = \{0\}$.

Moreover, according to Eq. (3.40), we obtain that

$$\mathbf{R}x = \begin{cases} \left(\frac{1}{2}-\frac{1}{4N}\right)x & \forall x \in \mathscr{S}_1 \\ \frac{1}{4}x & \forall x \in \mathscr{S}_2 \\ 0 & \forall x \in \mathscr{S}_3 \end{cases}$$

As a consequence, the non-null eigenvalues of $\mathbf{R}$ are $1/4$ and $(1/2-1/(4N))$ which implies that $\rho(\mathbf{R}) = 1/2-1/(4N)$. Hence, the convergence speed bound obtained by our general alternative approach developed here provides the true convergence speed for the Kempe's algorithm [43].

3.6.2 Comparison with Bénézit's algorithm

In [38], it has been shown that doing a multi-hop communication between sensors provides significant performance gain. However, the proposed algorithm relied on a single-variate algorithm. In order to ensure the convergence of this algorithm, the double-stochasticity of the matrix update is necessary which implies a feedback along the route. The feedback can suffer from link failure (due to high mobility in wireless networks). To counter-act this issue, Bénézit proposed to get rid of the feedback by using the Sum-Weight approach [42]. In this paper, the authors established a general convergence theorem close to ours. In contrast, they did not provide any result about convergence speed. It is worth noting that our convergence speed results can apply to the Bénézit's algorithm.

3.6.3 Comparison with the single-variate algorithms

In our analysis of standard gossip algorithms for averaging in Section 3.2, we assumed that the updates matrices were doubly-stochastic and showed that if they were only row-stochastic, the derived algorithms converged to a random value centered on x_{ave}. As we only assumed column-stochasticity in our analysis of Sum-Weight algorithms, let us see what our results tell us when the updates matrices are also row-stochastic.

Assumption 3.21. *In addition to Assumption 3.7, the matrices of $\mathscr{K}$ are row-stochastic*[10].

If the previous additional assumption holds, one can easily show that the weight sequence $\{w^k\}_{k>0}$ remain constant and equal to $\mathbb{1}$, i.e.,

$$\begin{aligned} &\forall k>0, && w^{k+1} = \mathbf{P}^{1,k+1} w^0 = \mathbf{P}^{1,k+1}\mathbb{1} = \mathbb{1} \\ &\text{and so} && x^{k+1} = s^{k+1} = \mathbf{K}_{\xi^{k+1}} x^k. \end{aligned}$$

Therefore, the single-variate algorithms with double-stochastic update matrices such as the Random Gossip [11], the Geographic Gossip [37] can completely be cast into the Sum-Weight framework. Moreover as $\forall k > 0$, $\Psi_1^k = \|x^0\|_2^2$ because all the weights stay equal to 1, the proposed results about $\{\Psi_2^k\}_{k>0}$ (that is Section 3.4.3) can be applied directly to the MSE for these algorithms.

Let us re-interpret the work of Boyd *et al.* [6] (especially their Section II) in the light of our results. In [6], it is stated that under doubly-stochastic update matrices, the MSE at time k is dominated by $\rho(\mathbb{E}[\mathbf{K}^{\mathrm{T}}\mathbf{K}]-(1/N)\mathbb{1}\mathbb{1}^{\mathrm{T}})^k$ using the same technique as in Section 3.2.2-b. In addition, they prove that the Random Gossip algorithm converges to 0 when k goes to infinity if

$$\rho\left(\mathbb{E}[\mathbf{K}]-\frac{1}{N}\mathbb{1}\mathbb{1}^{\mathrm{T}}\right)<1 \tag{3.41}$$

which is equivalent to Assumption 3.7c.

Furthermore, in [6, Section II-B], it is stated that the condition corresponding to Eq. (3.41) is only a sufficient condition and that the necessary and sufficient condition is the following one

$$\rho\left(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]-\mathbf{J}\right)<1 \tag{3.42}$$

which is exactly the same expression as in Lemma 3.14[11].

Moreover, according to [6, Eq. (19)] and Eq. (3.35), we know that the MSE at time k is upper bounded by $-\omega' k$ with $\omega' = -\log(\rho(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]-(1/N)\mathbb{1}\mathbb{1}^{\mathrm{T}})) > 0$. However, as mentioned in Section 3.4.3-c the condition $\omega' > 0$ seems only sufficient whereas $\omega > 0$ is a necessary and sufficient condition; furthermore, $\rho(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]\ \mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp})$ (*i.e.* $\mathrm{e}^{-\omega}$) is in general smaller than $\rho(\mathbb{E}[\mathbf{K}\otimes\mathbf{K}]-(1/N)\mathbb{1}\mathbb{1}^{\mathrm{T}})$ (*i.e.* $\mathrm{e}^{-\omega'}$). This accounts for our approach when analyzing convergence speed of gossip algorithms.

In Fig. 3.6, we display the empirical convergence slope, the associated lower-bound ω, and the bound given in [6] for the *Random Gossip* algorithm versus the number of sensors N. The proposed bound ω seems to fit much better than the one proposed in [6]. Actually, our proposed bound matches well the empirical slope.

[10] as they were column-stochastic with Assumption 3.7, this additional condition makes them doubly-stochastic.

[11] Indeed, as the vector v used in our formulation can be replaced with the left eigenvector corresponding to the eigenvalue 1 (see the proof of Lemma 3.14 for more details) which is proportional to $\mathbb{1}$ here due to the double-stochasticity of the update matrices

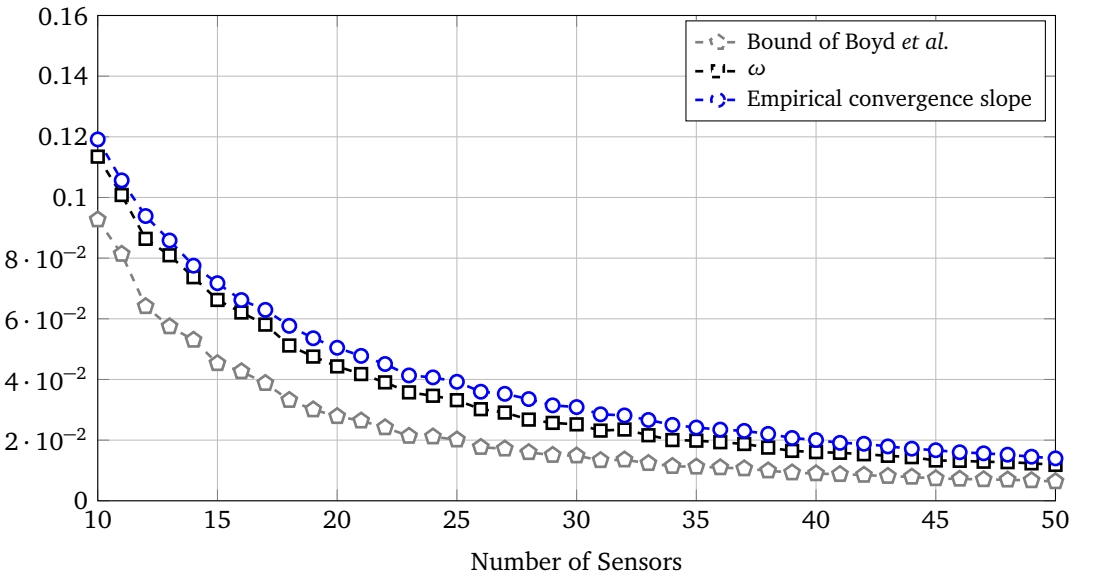


Figure 3.6: Empirical convergence slope of the Random Gossip, and associated lower bounds.

3.7 An application of averaging algorithms to cognitive radio

In this section, we will give an example of how averaging algorithms can be used in a cognitive radio context.

3.7.1 The problem of distributed spectrum sensing in cognitive radio networks

In some applications, agents of a WSN have to detect quickly the presence or absence of a signal of interest. For instance, one can mention the spectrum sensing in cognitive radio or the intrusion detection in military mobile ad-hoc networks. In order to make an accurate decision, these agents/nodes may have to cooperate with each other.

The traditional way to deal with a detection problem in a network consists in providing hard or soft detection decisions to a fusion center which makes a decision with the aggregated data and then transmits it to the agents of the network. This centralized approach is thus clearly sensitive to fusion center failure. Moreover, in the ad-hoc networks context, a fusion center election and a dedicated routing protocol have to be carried out which is costly in terms of overhead and time. Therefore, designing fully distributed decision algorithms is of great interest. Such algorithms rely on decision test functions and decision thresholds computed in a distributed way, *i.e.*only exchanging local data with neighbors.

Cooperative detection has recently received a lot of attention (see [52] and references therein). Nevertheless, most works assume the existence of a fusion center and finally focus on the design of operations done at each node in order to help the fusion center make the right decision. In the literature, only a few algorithms are fully distributed in the sense defined

above [53, 54, 55]. An important difference is that sensing and gossiping steps are alternated in [53, 54] whereas sensing steps come before gossiping steps in [55].

These algorithms are well adapted to time-varying environments but they suffer from difficulties in computing the threshold distributively. Indeed, in [53, 55], the threshold is chosen in the asymptotic regime and performances (especially false alarm probability) are not ensured in finite time. In [54], the threshold is chosen assuming the absence of diffusion/gossiping step. Hence, the threshold distributed computation remains an open issue.

We thus propose a new fully distributed signal decision algorithm based on our proposed algorithm, the *BWGossip* (see Section 3.5.1) where sensing steps are followed by gossiping steps and where the threshold is chosen adequately even in finite time. In addition, thanks to the separation of both steps, we are able to optimize their durations at the expense of less adaptivity compared to [53, 54].

3.7.2 Model

We consider a network of N nodes collaborating to detect the presence or absence of a signal. The received signal at time k on node i is y_i^k. We assume that the number of samples used for sensing is the same for all nodes and equal to N_s so for any sensor i we will write $y_i = [y_i^1, ..., y_i^{N_s}]^{\mathrm{T}}$ the vector of received data and $y = [y_1, ..., y_N]^{\mathrm{T}}$.

The signal to (potentially) detect is denoted by x_i^k at time k on node i and an additive noise can disturb the detection and is denoted by n_i^k at node i. We will model this noise at sensor i by a Gaussian random vector with zero mean and covariance matrix $\sigma_i^2\mathbf{I}$; formally, we have $n_i \sim \mathcal{N}(0, \sigma_i^2\mathbf{I})$ where $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ denote a Gaussian random vector with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$. We assume that the statistics of x_i and n_i are known at node i.

The binary hypothesis test of our problem can thus be written as follows

$$\begin{cases} \mathscr{H}_0: & \forall i, \quad y_i = n_i \\ \mathscr{H}_1: & \forall i, \quad y_i = x_i + n_i \end{cases} \tag{3.43}$$

and we want to build a test $\Lambda(y)$ whose goal is to enable us to separate the hypotheses so that if $\Lambda(y) \geq \delta$ then we decide that hypothesis $\mathscr{H}_1$ is valid and $\mathscr{H}_0$ elsewhere. We introduce two conditional probabilities that enable us to evaluate the performance of any binary test: i) the *probability of detection* P_{D}; and ii) the *false alarm probability* P_{FA} defined as:

$$P_{\mathrm{D}} \triangleq \mathbb{P}[\Lambda(y) > \delta | \mathscr{H}_1] \text{ and } P_{\mathrm{FA}} \triangleq \mathbb{P}[\Lambda(y) > \delta | \mathscr{H}_0]. \tag{3.44}$$

In the context of cognitive radio, it seems natural to ensure a fixed probability of detection as the secondary users may have non-disturbance requirements towards the primary users. Moreover, a high false alarm probability only implies that the secondary users do not use the white spaces while they could. Consequently, we want to minimize the false alarm probability for a fixed probability of detection which is the opposite of the standard Neyman-Pearson

criterion[12] (see [56, Chap. 2.2]).

Lemma 3.22. *In the case of a binary decision test with a constrained P_{D}, the optimal test is the Log-Likelihood Ratio (LLR) defined as*

$$\Lambda(y) \triangleq \log\left(\frac{p(y|\mathscr{H}_1)}{p(y|\mathscr{H}_0)}\right) \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \delta. \tag{3.45}$$

where $p(y|\mathscr{H})$ is the probability density of y given the tested hypothesis $\mathscr{H}$ and where δ is chosen such that the target probability of detection, denoted by $P_{\mathrm{D}}^{\mathrm{target}}$, is ensured.

Proof. The proof follows the approach developed in [56, Chap. 2.2] in the case of the Neyman-Pearson criterion.

Let us partition the received signal space $\mathscr{Z}$ into two subspaces : $\mathscr{Z}_0$ corresponding to the signals for which $\Lambda(y) < \delta$ (so for which we will decide $\mathscr{H}_0$) and $\mathscr{Z}_1$ corresponding to the signals for which $\Lambda(y) \geq \delta$ (we will decide $\mathscr{H}_1$). Now, let us consider the optimization problem

$$\begin{aligned} &\underset{\mathscr{Z}_1}{\text{minimize}}\ P_{\mathrm{FA}} \\ &\text{subject to } P_{\mathrm{D}}^{\mathrm{target}} - P_{\mathrm{D}} \leq 0 \end{aligned}$$

which we solve through the function F where λ is a Lagrange multiplier

$$\begin{aligned} F &= P_{\mathrm{FA}} + \lambda(P_{\mathrm{D}}^{\mathrm{target}} - P_{\mathrm{D}}) \\ &= \int_{\mathscr{Z}_1} p(y|\mathscr{H}_0)\mathrm{d}y + \lambda\left(P_{\mathrm{D}}^{\mathrm{target}} - \int_{\mathscr{Z}_1} p(y|\mathscr{H}_1)\mathrm{d}y\right) \\ &= \lambda P_{\mathrm{D}}^{\mathrm{target}} + \int_{\mathscr{Z}_1} (p(y|\mathscr{H}_0) - \lambda p(y|\mathscr{H}_1))\,\mathrm{d}y. \end{aligned}$$

Hence, for any $\lambda \geq 0$, the behavior (*i.e.*the choice of the partition of $\mathscr{Z}$ into $\mathscr{Z}_0$ and $\mathscr{Z}_1$) which minimizes F is "deciding $\mathscr{H}_1$ (*i.e.*being in $\mathscr{Z}_1$) if the term in the bracket is negative" *i.e.*

$$\frac{p(y|\mathscr{H}_0)}{p(y|\mathscr{H}_1)} < \delta.$$

As a conclusion, the optimal test is a Likelihood Ratio Test (LRT) as in the standard Neyman-Pearson approach. For the sake of simplicity we will work only on the classical LLR in the following. □

3.7.3 Review on centralized cooperative spectrum sensing

Before going further, we remind some important results about centralized cooperative spectrum sensing. We focus, on the one hand, on an energy-based detector (when the sought signal is unknown) and, on the other hand, on a training-based detector (when the sought signal is known and thus corresponds to a training sequence [54]).

[12] In the standard Neyman-Pearson criterion, one wants to maximize the probability of detection given a target false alarm probability.

3.7.3-a Energy-based detector

When the sought signal is unknown, it is usual to assume x_i is a zero-mean Gaussian vector with covariance matrix $\gamma_i^2\mathbf{I}$, *i.e.* $x_i \sim \mathcal{N}(0, \gamma_i^2\mathbf{I})$. Then, the Signal-to-Noise Ratio (SNR) at node i is equal to $\mathrm{SNR}_i \triangleq \gamma_i^2/\sigma_i^2$ and is assumed to be known at node i.

Assuming independence of the received signals at different nodes (this assumption is reasonable since even if the same signal is transmitted by the primary user, the random wireless channel leads to independent received signals between nodes), the test given in Eq. (3.45) can be decomposed as follows:

$$\Lambda(y) = \sum_{i=1}^{N} \Lambda_i(y_i)$$

with $\Lambda_i(y_i) = \log(p(y_i|\mathscr{H}_1)/p(y_i|\mathscr{H}_0))$.

As $x_i \sim \mathcal{N}(0, \gamma_i^2\mathbf{I})$ and $n_i \sim \mathcal{N}(0, \sigma_i^2\mathbf{I})$, we obtain the following test by removing the constant terms

$$\mathrm{T}(y) \triangleq \frac{1}{N}\sum_{i=1}^{N} \frac{\|y_i\|_2^2}{\gamma_i^2 + \sigma_i^2}\mathrm{SNR}_i \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta \tag{3.46}$$

where η must be chosen such that $\mathbb{P}[\mathrm{T}(y) > \eta|\mathscr{H}_1] = P_{\mathrm{D}}^{\text{target}}$.

In order to compute the threshold η, we need to exhibit the probability density of T under hypothesis $\mathscr{H}_1$. Unfortunately, due to the unequal SNRs between the sensors, T is not χ^2-distributed as in the standard energy detector. In [57], it is advocated that the density of T can be approximated with a Gamma distribution, denoted by $\Gamma(\kappa, \theta)$, whose the probability density function is equal to $g_{\kappa,\theta}$ defined by

$$g_{\kappa,\theta}(x) = \frac{1}{\Gamma(\kappa)\theta^\kappa} x^{\kappa-1}\mathrm{e}^{-x/\theta}, \quad x \geq 0, \tag{3.47}$$

and 0 otherwise. The mean of this distribution is $\kappa\theta$ and its variance is $\kappa\theta^2$, we thus choose κ and θ so that the mean and variance of the approximating Gamma distribution are the same as the ones of T. After some algebraic manipulations detailed in Appendix B.3, we obtain that $\mathrm{T} \approx \Gamma(\kappa_T, \theta_T)$ under hypothesis $\mathscr{H}_1$ with

$$\kappa_{\mathrm{T}} = \frac{NN_s}{2} \cdot \frac{\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2} \quad \text{and} \quad \theta_{\mathrm{T}} = \frac{2}{N} \cdot \frac{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i}. \tag{3.48}$$

One can then deduce that the optimal threshold η given the target probability of detection $P_{\mathrm{D}}^{\text{target}}$ is such that

$$P_{\mathrm{D}}^{\text{target}} = \mathbb{P}[\mathrm{T}(y) > \eta|\mathscr{H}_1] \approx 1 - G_{\kappa_{\mathrm{T}},\theta_{\mathrm{T}}}(\eta)$$

which leads to

$$\eta = G^{(-1)}_{\kappa_{\mathrm{T}},\theta_{\mathrm{T}}}(1 - P_{\mathrm{D}}^{\text{target}})$$

where $G_{\kappa,\theta}$ is the cumulative distribution function (cdf) of a Gamma distribution with parameters (κ, θ) and $G^{(-1)}_{\kappa,\theta}$ is its inverse.

Finally, we can derive the Receiver Operating Characteristic (ROC) of the test, that is the relation between the false alarm probability and the probability of detection,

$$P_{\mathrm{FA}} = 1 - G_{\kappa'_{\mathrm{T}},\theta'_{\mathrm{T}}}\left(G^{(-1)}_{\kappa_{\mathrm{T}},\theta_{\mathrm{T}}}(1-P_{\mathrm{D}})\right) \tag{3.49}$$

with

$$\kappa'_{\mathrm{T}} = \frac{N_s\left(\sum_{i=1}^{N}\frac{\mathrm{SNR}_i}{1+\mathrm{SNR}_i}\right)^2}{2\sum_{i=1}^{N}\left(\frac{\mathrm{SNR}_i}{1+\mathrm{SNR}_i}\right)^2} \text{ and } \theta'_{\mathrm{T}} = \frac{2\sum_{i=1}^{N}\left(\frac{\mathrm{SNR}_i}{1+\mathrm{SNR}_i}\right)^2}{\sum_{i=1}^{N}\frac{\mathrm{SNR}_i}{1+\mathrm{SNR}_i}}.$$

3.7.3-b Training-based detector

We now assume that each node i has the knowledge of the (possible) transmit signal x_i. Typically, the signal x_i may decomposed as $h_i x$ where h_i corresponds to the (known) channel fading between the node i and the sought transmitter and x is a (common) training sequence [54]. Here, the signal power is $\gamma_i^2 = \|x_i\|^2/N_s$.

Then the test given in Eq. (3.45) takes the following form

$$\mathrm{T}(y) \triangleq \frac{1}{N}\sum_{i=1}^{N}\frac{y_i^{\mathrm{T}}x_i}{\sigma_i^2} \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta. \tag{3.50}$$

As x_i is deterministic, T is Gaussian-distributed with mean m_{T} and variance ς_{T}^2 under hypothesis $\mathscr{H}_1$ given by

$$m_{\mathrm{T}} = N_s\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right) \text{ and } \varsigma_{\mathrm{T}}^2 = \frac{N_s}{N}\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right).$$

As a consequence, the obtained threshold is

$$\eta = \varsigma_{\mathrm{T}} Q^{(-1)}\left(P_{\mathrm{D}}^{\mathrm{target}}\right) + m_{\mathrm{T}}$$

where $Q^{(-1)}$ is the inverse of the Gaussian tail function.

3.7.4 Fully distributed spectrum sensing algorithms

Obviously, the tests described in Eqs. (3.46)-(3.50) are not computable since a node may not have the information from all the others. To overcome this problem, we propose to introduce a gossiping step in order to compute the involved averages. Indeed both previously derived tests can rewrite

$$\mathrm{T}(y) = \frac{1}{N}\sum_{i=1}^{N} t_i(y_i)$$

with

$$t_i(y_i) = \begin{cases} \|y_i\|_2^2\mathrm{SNR}_i/(\gamma_i^2+\sigma_i^2) & \text{in the case of energy detection} \\ y_i^{\mathrm{T}}x_i/\sigma_i^2 & \text{in the case of training-based detection.} \end{cases}$$

It is thus interesting to perform N_g gossipping steps after the N_s sensing steps in order to obtain a test close to the centralized one. Using a matrix formalism (see Section 3.2 for details), we have

$$\begin{bmatrix} T_1(y) \\ \vdots \\ T_N(y) \end{bmatrix} \triangleq \mathbf{P}^{1,N_g} \begin{bmatrix} t_1(y_1) \\ \vdots \\ t_N(y_N) \end{bmatrix}$$

where $T_i(y)$ is the final test function at node i, and where $\mathbf{P}^{1,N_g}$ corresponds to the considered gossiping algorithm matrix after N_g iterations.

It is thus easy to design fully distributed algorithms that estimate the test $\mathrm{T}(y)$. Actually, our main issue is to find a distributive way for computing a *good* threshold at any time, ensuring that the common target probability of detection $P_{\mathrm{D}}^{\mathrm{target}}$ is as close as possible at any step of the algorithm. Indeed, the decision is made before the convergence of the gossip algorithm and, assuming a primary user is present, T_i may be above the threshold whereas the gossip has not still converged to the consensus.

3.7.4-a Energy-based detector

When an energy-based detection is carried out, the final test function at node i is

$$T_i(y) = \sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \frac{\|y_j\|_2^2}{\gamma_j^2 + \sigma_j^2} \mathrm{SNR}_j \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta_i,$$

where η_i is the threshold at node i.

Once again, by assuming that T_i is well approximated by a Gamma distribution, using the same technique as in Appendix B.3 we obtain that under $\mathscr{H}_1$

$$\eta_i = G_{\kappa_i,\theta_i}^{(-1)}\left(1 - P_{\mathrm{D},i}^{\mathrm{target}}\right) \tag{3.51}$$

where $P_{\mathrm{D},i}^{\mathrm{target}}$ is the target detection probability at node i, and

$$\kappa_i = \frac{N_s}{2} \cdot \frac{\left(\sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \mathrm{SNR}_j\right)^2}{\sum_{j=1}^{N} \left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \mathrm{SNR}_j^2} \quad \text{and} \quad \theta_i = 2 \cdot \frac{\sum_{j=1}^{N} \left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \mathrm{SNR}_j^2}{\sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \mathrm{SNR}_j}. \tag{3.52}$$

Then, we obtain that the ROC is equal to

$$P_{\mathrm{FA},i} = 1 - G_{\kappa_i',\theta_i'}\left(G_{\kappa_i,\theta_i}^{(-1)}\left(1 - P_{\mathrm{D},i}\right)\right) \tag{3.53}$$

with

$$\kappa_i' = \frac{N_s}{2} \cdot \frac{\left(\sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \frac{\mathrm{SNR}_j}{1+\mathrm{SNR}_j}\right)^2}{\sum_{j=1}^{N} \left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \left(\frac{\mathrm{SNR}_j}{1+\mathrm{SNR}_j}\right)^2} \quad \text{and} \quad \theta_i' = 2 \cdot \frac{\sum_{j=1}^{N} \left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \left(\frac{\mathrm{SNR}_j}{1+\mathrm{SNR}_j}\right)^2}{\sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \frac{\mathrm{SNR}_j}{1+\mathrm{SNR}_j}}.$$

We remark that our ROC curve depends on the gossip algorithm. Unfortunately, the terms involving $(\mathbf{P}_{i,j}^{1,N_g})^2$ in κ_i' prevent us from obtaining the threshold η_i in a distributed way at node

i for ensuring the probability of detection $P_{\mathrm{D},i}^{\mathrm{target}}$. To overcome this issue, we propose hereafter two approaches.

Approach 1: distributed test with knowledge of N. Actually, in the centralized setup, the threshold depends on the average of the SNR and the squared SNR through Eq.(3.48). A simple idea is to replace these exact averages with the averages obtained thanks to the considered gossip algorithm. It is clear that if N_g is large enough, the obtained thresholds will be close to those of the centralized case and also to those described in Eq. (3.52). As a consequence, the new threshold is

$$\eta_i^{(1)} = G^{(-1)}_{\kappa_i^{(1)},\theta_i^{(1)}}\left(1-P_{\mathrm{D},i}^{\mathrm{target}}\right) \tag{3.54}$$

with

$$\kappa_i^{(1)} = \frac{NN_s}{2}\cdot\frac{\left(\sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g}\mathrm{SNR}_j\right)^2}{\sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g}\mathrm{SNR}_j^2} \text{ and } \theta_i^{(1)} = \frac{2}{N}\cdot\frac{\sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g}\mathrm{SNR}_j^2}{\sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g}\mathrm{SNR}_j}.$$

This algorithm is still not fully distributed since the knowledge of the number of nodes is required but the threshold only depends on the cdf of a Gamma distribution (which can be tabulated to avoid further complexity) with parameters depending only on terms that can be computed by gossipping the vector of the SNRs vector and the squared SNRs vector along with the individual tests.

Unfortunately, the target probability of detection is not ensured since the real probability of detection, denoted by $P_{\mathrm{D},i}^{(1)}$, is given by

$$P_{\mathrm{D},i}^{(1)} = 1-G_{\kappa_i,\theta_i}(G^{(-1)}_{\kappa_i^{(1)},\theta_i^{(1)}}(1-P_{\mathrm{D},i}^{\mathrm{target}})).$$

In contrast, we prove that the ROC curve is the following one

$$P_{\mathrm{FA},i}^{(1)} = 1-G_{\kappa_i',\theta_i'}(G^{(-1)}_{\kappa_i,\theta_i}(1-P_{\mathrm{D},i}^{(1)}))$$

which is the same as in Eq. (3.53). Consequently, the ROC curve is not degraded due to our approximate threshold. In addition, the operating point in the ROC curve can not fixed a priori.

Approach 2: fully distributed test. In this approach, the knowledge of the number of nodes will not be required anymore. As seen in Section 3.5.4, our proposed algorithm, the *BWGossip*, is able to perform the estimation of the sum jointly with the estimation of the average.

We define

$$\mathbf{M}^{1,N_g} \triangleq \mathrm{diag}\left(\frac{1}{\mathbf{P}^{1,N_g}\mathbb{1}}\right)\mathbf{P}^{1,N_g} \text{ and } \mathbf{S}^{1,N_g} \triangleq \mathrm{diag}\left(\frac{1}{\mathbf{P}^{1,N_g}e_1}\right)\mathbf{P}^{1,N_g} \tag{3.55}$$

so that $\mathbf{M}^{1,N_g}x^0$ (resp. $\mathbf{S}^{1,N_g}x^0$) is the estimate of the average (resp. the sum) of x^0 obtained by a Sum-Weight-based algorithm (as the *BWGossip*).

In this setup, before gossipping, the two auxiliary variables of each node must be initialized to match $\mathbb{1}$ and e_1 respectively. For the first auxiliary variable, each node is initialized to 1. For the second one, only the first node is initialized to 1 whereas the others to 0. In cognitive radio context, this means the first node is the secondary user launching the sensing, *i.e.*which wants to access the medium.

Recalling Eq. (3.48), we see that eliminate the factor N in the coefficients κ_{T} and θ_{T}: i) in κ_{T}, by replacing the SNRs averages by the sum of the SNRs; and ii) in θ_{T}, by replacing the SNRs average in the denominator by the sum of the SNRs. Consequently, replacing the involved sums and averages by estimates found through gossip, we find

$$\eta_i^{(2)} = G^{(-1)}_{\kappa_i^{(2)},\theta_i^{(2)}}\left(1 - P_{\mathrm{D},i}^{\mathrm{target}}\right) \tag{3.56}$$

with

$$\kappa_i^{(2)} = \frac{N_s}{2} \cdot \frac{\left(\sum_{j=1}^N \mathbf{S}_{i,j}^{1,N_g} \mathrm{SNR}_j\right)^2}{\sum_{j=1}^N \mathbf{S}_{i,j}^{1,N_g} \mathrm{SNR}_j^2} \quad \text{and} \quad \theta_i^{(2)} = 2 \cdot \frac{\sum_{j=1}^N \mathbf{M}_{i,j}^{1,N_g} \mathrm{SNR}_j^2}{\sum_{j=1}^N \mathbf{S}_{i,j}^{1,N_g} \mathrm{SNR}_j}.$$

The algorithm is now fully distributed since even the number of nodes is not required. Once again, the new threshold does not ensure the target probability of detection, and the ROC curve is still described by Eq. (3.53) but $\mathbf{P}$ is replaced by $\mathbf{M}$ given by Eq. (3.55).

Finally, for both approaches, the ROC curves converge to the ROC curve related to the centralized case when N_g is large enough since the parameters κ_i, κ_i', θ_i and θ_i' all converge to those of the centralized case.

3.7.4-b Training-based detector

In the case of the training-based detector, the test function at node i after N_g gossiping iterations is

$$T_i(y) = \sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g} \frac{y_j^{\mathrm{T}} x_j}{\sigma_j^2} \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta_i,$$

As T_i is Gaussian distributed with mean m_i and variance ς_i^2 under $\mathscr{H}_1$, we have

$$\eta_i = \varsigma_i Q^{(-1)}\left(P_{\mathrm{D},i}^{\mathrm{target}}\right) + m_i$$

and

$$m_i = N_s \sum_{j=1}^N \mathbf{P}_{i,j}^{1,N_g} \mathrm{SNR}_j \text{ and } \quad \varsigma_i^2 = N_s \sum_{j=1}^N \left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \mathrm{SNR}_j.$$

Once again, this algorithm can not be computed in a distributed way due to the presence of the terms $(\mathbf{P}_{i,j}^{1,N_g})^2$ in the variance. To overcome this issue, the previous proposed approaches can be applied straightforwardly.

3.7.5 Numerical illustrations

Except otherwise stated, an energy-based detector is carried out with $N_s = N_g = 64$ and $P_{\mathrm{D},i}^{\mathrm{target}} = 0.99$ for all i, and performance are averaged over RGGs with $N = 10$ nodes. The SNRs at each node are exponentially-distributed with mean $\overline{\mathrm{SNR}}$. Only performance for the node exhibiting the smallest SNR realization will be plotted.

Hereafter, we test the four following algorithm configurations: i) the centralized one; ii) the *Random Gossip* with centralized threshold (see Eq. (3.51)); iii) the *Random Gossip* with the threshold of approach 1 (see Eq. 3.54); and iv) the *BWGossip* with the threshold of approach 2 (see Eq. (3.56)).

In Fig. 3.7, we plot the ROC curve for the above-mentioned algorithm configurations. We remark that the ROC curves (which only depends on the gossipping) are very close to each other. In addition, when the same gossip algorithm is used, the ROC curve is identical regardless of the threshold technique computation.

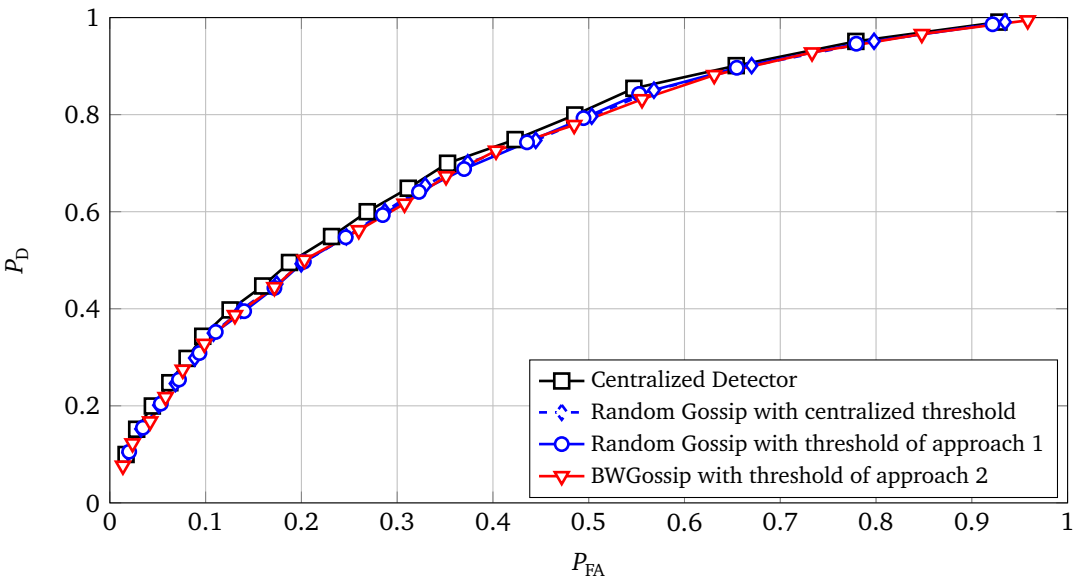


Figure 3.7: Receiver Operating Characteristic curve (P_{D} vs. P_{FA}).

In Figs. 3.8 and 3.9, we display empirical P_{FA} and P_{D} versus i) $\overline{\mathrm{SNR}}$ for $N_s = N_g = 64$; and ii) N_s for $N_t = N_s + N_g = 128$ fixed with mean SNR $-5dB$. The Random Gossip with centralized threshold performs almost as well as the centralized detector which implies that gossipping the personal tests instead of having a fusion center gathering and summing the sensors data is not very costly, thus the losses in performance are mainly due to the threshold computation. We remark that when the threshold is computed through approach one, the false alarm probability is lower than in the centralized case however the target probability of detection is not ensured especially when the number of gossipping steps is small (N_s close to 128). In the case of BWGossip with approach 2, the loss in false alarm probability is reasonable while its probability of detection is higher than the target one, it has thus very good performances. We also remark

the sensing-to-gossipping ratio that offers the best performance is common and close to 1 for all algorithms.

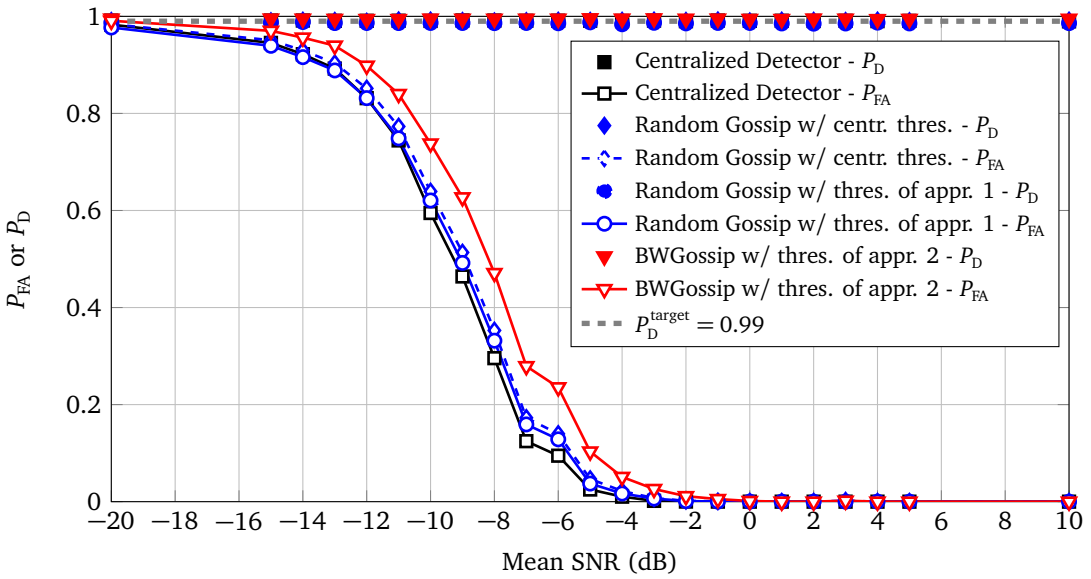


Figure 3.8: P_{FA} and P_D versus $\overline{\text{SNR}}$.

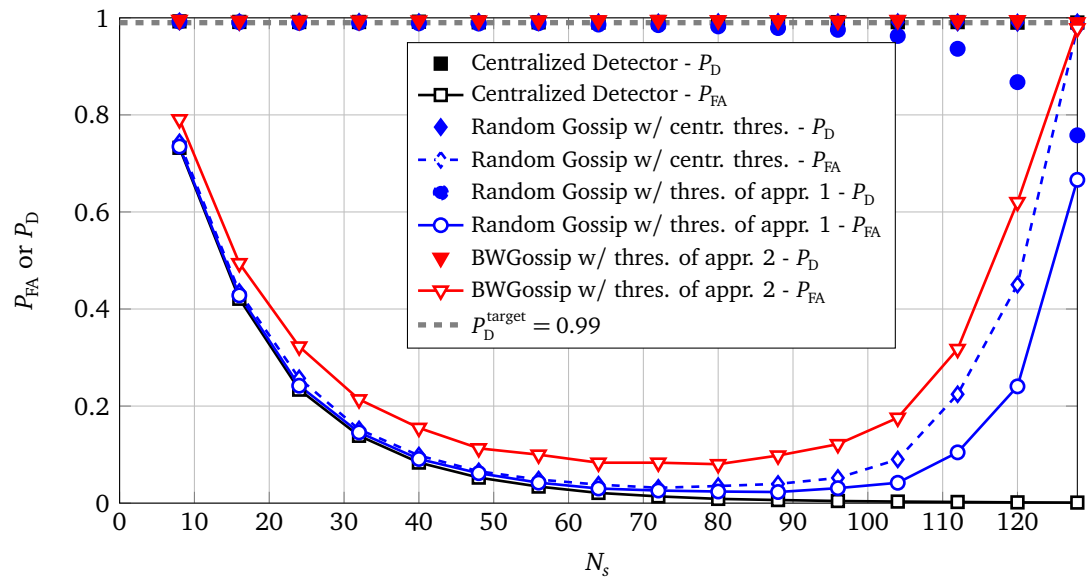


Figure 3.9: P_{FA} and P_D versus N_s.

Finally, in Fig. 3.10, we compare our training-based algorithms to the *Diffusion LMS* described in [54]. Notice that the sensing and gossipping steps are mixed for the diffusion LMS. We remark that our proposed algorithms outperform the diffusion LMS. Actually, our block processing for the sensing step is much more efficient that the adaptive LMS one in [54]. Moreover, unlike diffusion LMS, our algorithms are asynchronous which simplifies the net-

work management, and the threshold is chosen much more adequately.

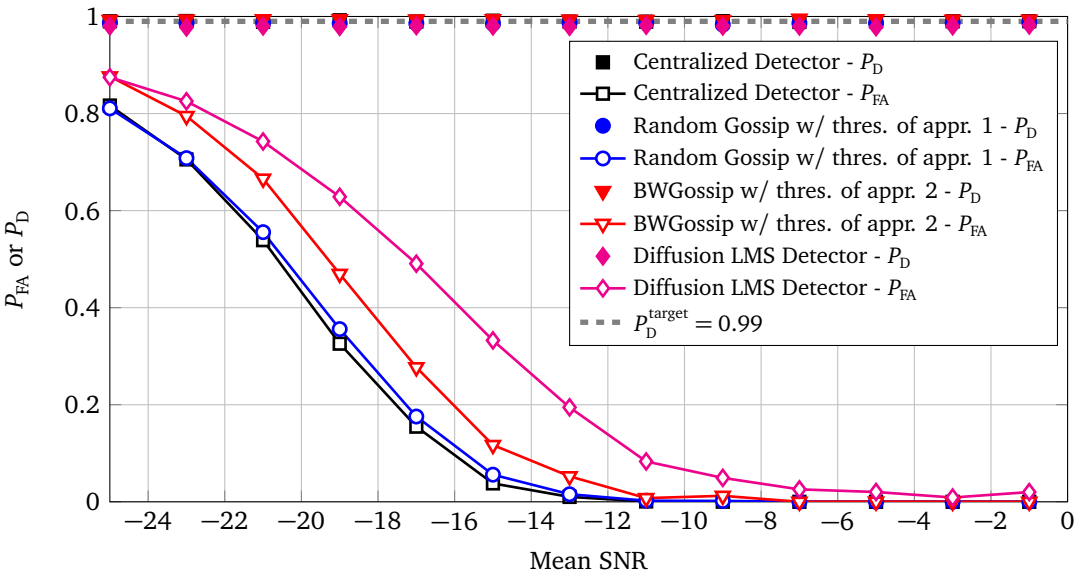


Figure 3.10: P_{FA} and P_D versus $\overline{SNR}$ for proposed algorithms and *Diffusion LMS*.

As mentioned before, one of the advantages of cooperation between secondary users is to solve the so-called *Hidden Terminal problem* shown in Fig. 3.11. In this figure, we see a cognitive radio network consisting of two Primary Users (PU) and five secondary users (SU). The primary user PU1 may contact PU2 in which case secondary user must not interfere, the links with associated long-term SNR are represented in dashed lines. In order to efficiently detect a communication from PU1, the secondary users may communicate through dedicated links represented by green lines. The *Hidden Terminal problem* problem occurs when a secondary user (SU1 in our case) is not able to detect the emitting primary user (PU1) because of a bad long-term SNR due to some obstacle or large distance; but can perturb another primary user (PU2) if it emits.

In Fig. 3.12, the four proposed algorithms of the previous section have been evaluated on the hidden terminal practical configuration described in Fig. 3.11. We represent the probability of detection and false alarm probability of the secondary user #1 versus $N_t = N_s + N_g$ with $N_s = N_g$. The centralized detector is obviously very efficient as our user of interest then benefits from the measurements of all the network and thus from the sensors close to the emitting primary user. When gossipping, the hidden secondary user is in general poorly connected to to the other as represented in Fig. 3.11. Thus, asynchronous gossip-based algorithms have quit poor performance even in the centralized threshold case. As seen before, Approach 1 does not ensure the target probability of detection and thus cannot be used in a practical setting. Approach 2 with BWGossip seems to ensure our target probability of detection in most cases at the price of a higher false alarm probability. We can conclude that our most advanced algorithm is enable to distributively detect the hidden terminal quite quickly.

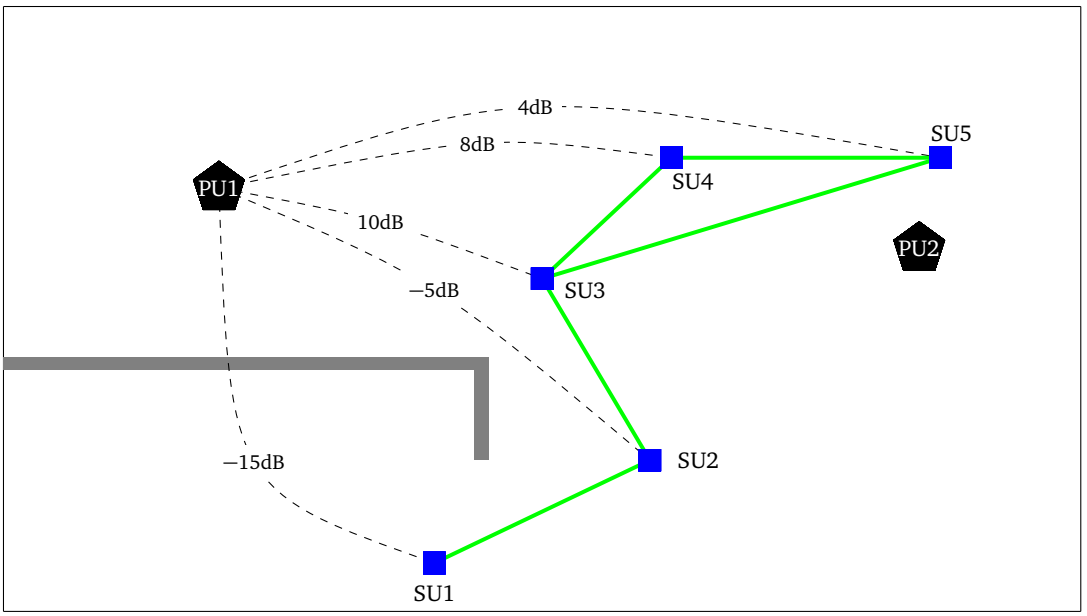


Figure 3.11: Hidden terminal configuration

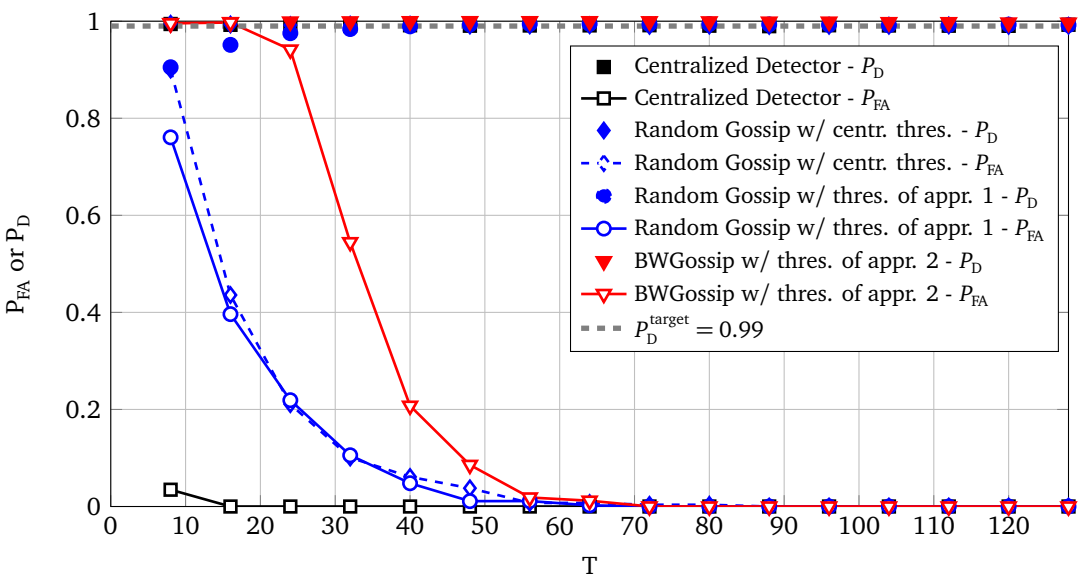


Figure 3.12: P_{FA} and P_D versus T for the hidden terminal problem of Fig. 3.11.

3.8 Conclusion

In this chapter, we proved very general results about convergence and convergence speed of average gossip algorithms over WSNs. The Sum-Weight framework enabled us to prove more general convergence properties and to derive a very efficient feedback-free broadcast algorithm which significantly outperforms the existing ones. In addition, we showed that our results applied to any standard averaging algorithm and that our derived speed bounds were in

general tighter than the ones in the literature. Finally, we applied our results to the problem of distributed spectrum in cognitive radio networks and enabled us to design a fully-distributed spectrum sensing algorithm.

In the next chapter, we will consider the problem of distributed optimization.

This work has led to the following publications:

J2 **F. Iutzeler**, P. Ciblat, and W. Hachem, "Analysis of Sum-Weight-like algorithms for averaging in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2802–2814, June 2013.

C3 **F. Iutzeler** and P. Ciblat, "Fully-distributed spectrum sensing: application to cognitive radio," in *European Signal Processing Conference (EUSIPCO)*, September 2013.

C2 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "New broadcast based distributed averaging algorithm over wireless sensor networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012.

N2 **F. Iutzeler** and P. Ciblat, "Sondage de spectre coopératif totalement distribué: application à la radio cognitive," in *Colloque GRETSI*, September 2013.

CHAPTER 4

DISTRIBUTED OPTIMIZATION

In this chapter, we focus on optimization algorithms over networks. After motivating our choices, we design and prove the convergence of an asynchronous distributed optimization algorithm.

4.1 Introduction

The recent years have seen a dramatic increase in the quantity of data available over the Web and in the computational abilities of any electronic device. The classical model of statistical inference where a single computation unit has to infer behaviors on a small data set is thus inadequate for such a setup. To take into account these recent developments [58], it is interesting to consider a system where:

- a large quantity of data is spread over many (physically) distant machines;
- these machines can perform costly computations;
- they need to communicate to infer a global behavior as their own dataset (although large) might not be representative;
- for evident congestion reasons, local, peer-to-peer communications must be preferred over a centralized scheme.

A large majority of data processing algorithms rely on the minimization of a well-designed cost function that takes into account the whole dataset. We will focus in this chapter on convex optimization, that is when the cost function is convex. $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a *convex* function if and only if its *epigraph*

$$\mathbf{epi}\, f = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq t\}$$

is a convex set; in addition, f will be *closed* if its epigraph is. Also, the optimal points will necessarily be in the effective *domain* of f

$$\mathbf{dom}\, f = \{x \ \in \mathbb{R}^n : f(x) < +\infty\}$$

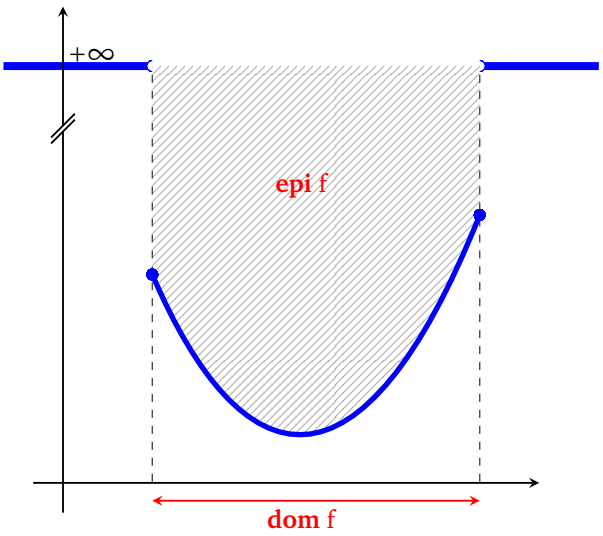


Figure 4.1: The epigraph and domain of a convex function.

i.e. the set of points for which f takes on finite values as represented on Fig. 4.1; if $\mathbf{dom}\, f \neq \emptyset$, f is *proper*.

The performance of these algorithms depend on i) the choice of the cost function, which mostly depends on the problem; and ii) the optimization algorithm used for minimizing it, that changes how the function is used in the iterations of the minimization.

In order to perform efficient data processing on big data networks, one has to design distributed optimization algorithms. Mathematically, a network of agents represented by its graph $\mathcal{G} = (V, E)$ is seeking to distributively solve the following optimization problem.

Problem 4.1.

$$\min_{x \in \mathbb{R}} \quad f(x) \triangleq \sum_{i \in V} f_i(x)$$

where f_i is a convex real cost function known only by agent i.

Function f_i can be interpreted as the price paid by agent i when the global network state is x. In the most general case the optimization variable x lives in any Euclidian space but for the sake of clarity we will work in $\mathbb{R}$ along this chapter. The goal of this chapter is thus to design optimization solving Problem 4.1 with the following particularities:

- the agents may only perform local updates, *i.e.* they update their private estimate using only their private cost function, indeed the global cost function is nowhere available;
- the agents can only communicate their variables locally using the edges of the underlying communication graph.

For these reasons, each agent will update its own variable x_i only according to its cost function f_i, it is thus useful to write Problem 4.1 making apparent the nodes variables and adding an equality (or *consensus*) constraint.

Problem 4.2.

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^N} & F(x) \triangleq \sum_{i\in V} f_i(x_i) \\ \textit{subject to} & x_1 = x_2 = \cdots = x_N \end{array}$$

A function that writes as a sum of functions over scalar components, as F, will be said *separable*.

As each sensor cannot know all $\{x_i\}_{i=1,\dots,N}$ and $\{f_i\}_{i=1,\dots,N}$, any distributed optimization has to feature two steps: i) a local variable update where one or more sensors update their local variables using their local cost function; and ii) a consensus step where some sensors exchange scalars with their neighbors in order to reach a consensus. The problem of reaching consensus with local exchanges (step ii) has been studied in details in Chapters 2 and 3. In the literature, two main classes of optimization algorithms exist for solving Problem 4.1 (or related ones). They differ from how a node makes use of its local function:

- the methods where the i-th node uses *local* properties of f_i at x_i as the gradient (or subgradient) introduced as a classical distributed computing problem in [45, Chap. 7.5] (based on [59, 60]) and refined in [61, 62, 63, 64]. One can also mention Nesterov-like methods [65, 66], or Newton-like methods [67].
- the methods where the nodes use their *whole* individual function through an argmin step for example. The most celebrated algorithm for distributed optimization is the *Alternating Direction Method of Multipliers (ADMM)*, popularized by the monograph [68], which uses a regularized argmin of the functions at each iteration. It was demonstrated to be particularly suited to graph-based (local) communications in [69, 70].

Standard algorithms are generally *synchronous* that is to say all agents are supposed to complete their local computations synchronously at each tick of some global clock, and then synchronously merge their local results. However, in many situations, one faces variable sizes of the local data sets along with heterogeneous computational abilities of the machines. Synchronism then becomes a burden as the global convergence rate is expected to depend on the local computation times of the slowest agents. In addition, we saw that in the previous chapters that synchronism in the communications can result in collisions and network congestions. It is thus crucial to introduce asynchronous methods which allow the estimates to be updated in a non-coordinated fashion, rather than all together or in some frozen order.

This chapter is organized as follows. First, Section 4.2 describes existing synchronous and asynchronous *first order* methods for distributed optimization. These methods are ultimately based on first-order derivatives of the cost functions and enable us to oversee some local optimization methods. Then, in Section 4.3, we introduce *proximal* methods which constitute a large majority of the optimization methods based on the whole functions. We put the focus on the ADMM algorithm and show how it can be used to solve a distributed optimization problem. Section 4.4 then recalls the basics of *monotone operators* theory which enables us to elegantly analyze proximal algorithms in general and the ADMM in particular. Finally, in Section 4.5 we introduce our new asynchronous ADMM-based algorithm for distributed optimization, we prove its convergence using monotone operator theory, and illustrate its performances in Sec-

tion 4.6.

4.2 First order methods

In this section, we will review some simple algorithms that distributively solve Problem 4.2 using local properties of the cost functions. We consider the case where the nodes update their value by computing a gradient of its cost function then exchange with each other in order to reach the wanted consensus. We will focus here on the consequences of these exchanges (or gossip) on the convergence properties of gradient algorithm.

These results are based on [59, 60, 62, 63] and [71].

4.2.1 Model

As previously mentioned, these algorithms are based on two steps: i) a gradient descent; then ii) a gossip step. We will thus make explicit the two steps independently and then put them together along with classical assumptions from the literature.

4.2.1-a Step 1: Gradient descent

In first order methods, every node i makes use of its individual function f_i through a *gradient descent* step. Assuming f_i is differentiable, the update equation reads:

$$x_i^{k+1} = x_i^k - \gamma^k \nabla f_i(x_i^k) \tag{4.1}$$

where $\nabla f_i(x_i)$ is the gradient of f_i at point x_i and $\{\gamma^k\}_{k>0}$ is a sequence of positive coefficients. For notational simplicity, we shall assume that the $\{f_i\}_{i=1,\dots,M}$ are $\mathbb{R} \to \mathbb{R}$ in which case the gradients boil down to derivatives[1]. Fig. 4.2 gives a representation of a gradient step with $\gamma^k = 1$.

If f_i has a unique minimum, then f_i' is non-decreasing and null at the point $x_i^\star$ where f_i is minimal. Then, it is easy to see that if $x_i^k > x_i^\star$, then $x_i^{k+1} < x_i^k$ and conversely. The sequence $\{x_i^k\}_{k>0}$ produced by Eq. (4.1) then converges to $x_i^\star$ under some assumptions on f_i and $\{\gamma_i^k\}_{k>0}$ (see [72, Chap. 9] for details).

4.2.1-b Step 2: Gossipping

Let $\mathcal{G} = (V, E)$ be a network of nodes wanting to solve Problem 4.1 with first order methods. They need to maintain a common value $x \in \mathbb{R}$ and compute $f'(x) = \sum_{i \in V} f_i'(x)$ which is very difficult to implement without the presence of a fusion center. One way to overcome this problem is to compute this gradient along an Hamiltonian cycle (a cycle that visits exactly once every vertex) which leads to *incremental* gradient methods (see [73] and references therein for details). However, this method needs the creation of an Hamiltonian cycle which can be costly

[1] This avoids the use of Kronecker products in the equations related with the gossipping.

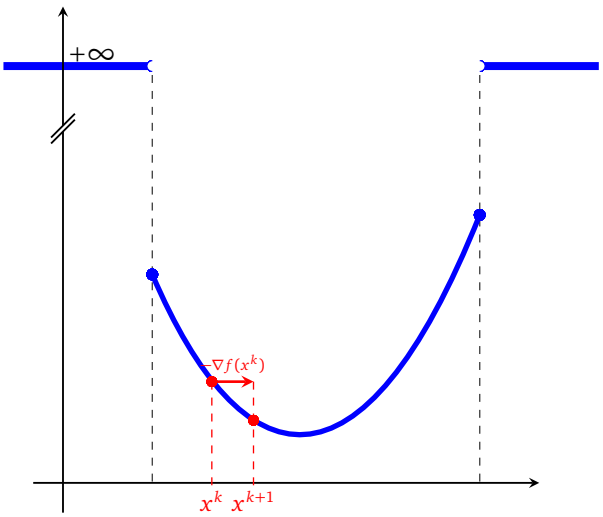


Figure 4.2: Illustration of a gradient descent step.

and implies the presence of an acting fusion center; furthermore, the resulting algorithm is quite sensitive to link failures.

We will thus consider a more distributed approach based on Problem 4.2 where the network performs a gradient descent on F (*i.e.* each sensor i performs a gradient descent on f_i) and the sensors exchange their estimates with their neighbors by an average gossip step as in Eq. (3.1):

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} x^k$$

where the process $\{\mathbf{K}_{\xi^k}\}_{k>0}$, valued in the set $\mathscr{K} = \{\mathbf{K}_i\}_{i=1,\dots,M}$, is i.i.d..

4.2.1-c Distributed gradient algorithms

Combining these two steps, we get the following algorithm where the word asynchronous means here that the communications between the agents are asynchronous as in the previous chapter; the gradient steps of the agents still have to be performed synchronously. As in Section 3.2, the process $\{\mathbf{K}_{\xi^k}\}_{k>0}$ of the communication matrices, valued in the set $\mathscr{K} = \{\mathbf{K}_i\}_{i=1,\dots,M}$, is i.i.d..

Recalling the definition of F in Prob. 4.2, the two steps can be merged into a single equation as

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} \left(x^k - \gamma^k \nabla F(x^k) \right) \tag{4.2}$$

or with matrices products

$$x^k = \mathbf{K}_{\xi^k} \mathbf{K}_{\xi^{k-1}} \dots \mathbf{K}_{\xi^1} x^0 - \sum_{\ell=1}^{k-1} \gamma^\ell \mathbf{K}_{\xi^k} \mathbf{K}_{\xi^{k-1}} \dots \mathbf{K}_{\xi^{\ell+1}} \nabla F(x^\ell). \tag{4.3}$$

In the next sections, we will see how gossipping enables this algorithm to converge to a consensus minimizing the function f. For the sake of clarity, we shall assume first that all the

Asynchronous distributed gradient algorithm

At each clock tick k:

- every sensor i performs a gradient descent on its cost function f_i:

$$\tilde{x}_i^{k+1} = x_i^k - \gamma^k f_i'(x_i^k)$$

- the sensors average their value using matrix $\mathbf{K}_{\xi^k}$:

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} \tilde{x}^{k+1}$$

---

matrices $\{\mathbf{K}_{\xi^k}\}_{k>0}$ are equal to a given matrix $\mathbf{K}$. This corresponds in practice to a *synchronous* gossipping scheme as in Section 3.2.2-a. The asynchronous scheme where the process $\{\mathbf{K}_{\xi^k}\}_{k>0}$ is i.i.d. will be considered as a second step.

Let us start by putting some general assumptions on the cost functions $\{f_i\}_{i=1,\dots,N}$, the stepsizes and the matrix $\mathbf{K}$.

Assumption 4.3. *The functions $\{f_i\}_{i=1,\dots,N}$ verify:*

i) The functions $\{f_i\}_{i=1,\dots,N}$ are convex and differentiable;

ii) The function f attains its infimum and the set of its minimizers $\mathscr{M}$ is compact;

iii) Their derivatives $\{f_i'\}_{i=1,\dots,N}$ are L-Lipschitz continuous and $\forall x, |f_i'(x)| \leq C$.

Assumption 4.4. *The positive sequence of stepsizes $\{\gamma^k\}_{k>0}$ verifies:*

i) $\sum_{k>0} \gamma^k = +\infty$;

ii) $\sum_{k>0} (\gamma^k)^2 < \infty$;

iii) $\gamma^{k+1}/\gamma^k \longrightarrow_{k\to\infty} 1$.

Assumption 4.5. *The matrix $\mathbf{K}$ is doubly stochastic with a positive diagonal and primitive.*

4.2.2 Convergence of the Synchronous Distributed gradient algorithm

As in Section 3.2.2-a, let us first review synchronous distributed gradient algorithms. Let $\mathbf{K}$ be the gossip matrix that models the exchanges at each iteration, the associated *distributed gradient* algorithm is presented below.

As before, the two steps can be rewritten in a single equation:

$$x^{k+1} = \mathbf{K}\big(x^k - \gamma^k \nabla F(x^k)\big). \tag{4.4}$$

We also have

$$x^k = (\mathbf{K})^k x^0 - \sum_{\ell=1}^{k-1} \gamma^\ell (\mathbf{K})^{k-\ell} \nabla F(x^\ell). \tag{4.5}$$

We will prove the convergence of this algorithm under the above stated conditions. The convergence proof will follow two main steps: i) we will prove that this sequence converges to a consensus; and ii) we will show that the value of the consensus is $x^\star$ such that $\nabla f(x^\star) = 0$.

Synchronous Distributed gradient algorithm

At each clock tick k:

- every sensor i performs a gradient descent on its cost function f_i:

$$\tilde{x}_i^{k+1} = x_i^k - \gamma^k f_i'(x_i^k)$$

- the sensors average their value using matrix $\mathbf{K}$:

$$x^{k+1} = \mathbf{K}\tilde{x}^{k+1}$$

4.2.2-a Convergence to a consensus

Recalling Eq. (4.5), we have

$$\begin{aligned}
\left\|\mathbf{J}^\perp x^{k+1}\right\|_2 &= \left\|\mathbf{J}^\perp\left((\mathbf{K})^{k+1}x^0 - \sum_{\ell=1}^{k}\gamma^\ell(\mathbf{K})^{k+1-\ell}\nabla F(x^\ell)\right)\right\|_2 \\
&= \left\|\left(\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp\right)^{k+1}x^0 - \sum_{\ell=1}^{k}\gamma^\ell(\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp)^{k+1-\ell}\nabla F(x^\ell)\right\|_2 \\
&\leq \left\|\left(\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp\right)^{k+1}x^0\right\|_2 + \sum_{\ell=1}^{k}\gamma^\ell\left\|(\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp)^{k+1-\ell}\nabla F(x^\ell)\right\|_2 \\
&\leq |||\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp|||_2^{k+1}\|x^0\|_2 + \sum_{\ell=1}^{k}\gamma^\ell|||\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp|||_2^{k+1-l}\|\nabla F(x^\ell)\|_2
\end{aligned}$$

where the first equality comes from the fact that $\mathbf{J}^\perp\mathbf{K} = \mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp$ as $\mathbf{K}$ is row-stochastic from Assumption 4.5. The second inequality comes from the induced norm inequality for matrices. Furthermore, the conditions of Theorem 3.4 are satisfied so $\sigma \triangleq |||\mathbf{J}^\perp\mathbf{K}\mathbf{J}^\perp|||_2 < 1$. We thus have

$$\left\|\mathbf{J}^\perp x^{k+1}\right\|_2 \leq (\sigma)^{k+1}\|x^0\|_2 + \gamma^k\sqrt{N}C\sum_{\ell=1}^{k}\frac{\gamma^\ell}{\gamma^k}(\sigma)^{k+1-\ell}$$

where the first term of the RHS goes to zero as $k \to \infty$; the second term is the product of γ^k and $\sqrt{N}C\sum_{\ell=1}^{k}\gamma^\ell/\gamma^k(\sigma)^{k+1-\ell}$. One can find in [74, Part 1, Pb. 178] that as i) $\gamma^{k+1}/\gamma^k \to 1$ from Assumption 4.4; and ii) $\sigma < 1$ as seen above; $\sum_{\ell=1}^{k}\gamma^\ell/\gamma^k(\sigma)^{k+1-\ell}$ converges. Hence, we have proved that

$$\left\|\mathbf{J}^\perp x^{k+1}\right\|_2 = \mathcal{O}\left(\gamma^k\right). \tag{4.6}$$

This implies that $\{\left\|\mathbf{J}^\perp x^{k+1}\right\|_2\}_{k>0}$ decreases to zero at the same speed as $\{\gamma^k\}_{k>0}$ hence the sensors variables go to a consensus. We will now study the value of this consensus.

4.2.2-b Consensus value

First, recalling Eq. (4.4) and using the identity $\mathbf{JK} = \mathbf{J}$ for any column-stochastic matrix $\mathbf{K}$, we observe that

$$\mathbf{J}x^{k+1} = \mathbf{J}x^k - \gamma^k \mathbf{J}\nabla F(x^k) \tag{4.7}$$

is *almost* a gradient descent of the sequence $\{\bar{x}^k\}_{k>0} \triangleq \{1/N\mathbb{1}^{\mathrm{T}}x^k\}_{k>0}$ (it would have been a gradient descent if the sequence $\bar{x}^k$ were described by the iterations $\bar{x}^{k+1} = \bar{x}^k - \gamma^k f'(\bar{x}^k) = \bar{x}^k - \gamma^k N(1/N\mathbb{1}^{\mathrm{T}}\nabla F(\bar{x}^k\mathbb{1}))$). Equation (4.7) can be rewritten as

$$\bar{x}^{k+1} = \bar{x}^k - \gamma^k \left(1/N\mathbb{1}^{\mathrm{T}}\nabla F\left(x^k\right)\right). \tag{4.8}$$

Let us take $x \leq x'$, as f is differentiable, the mean value theorem states that there is $x \leq c \leq x'$ such that

$$f(x') - f(x) = f'(c)(x' - x)$$

and as f is convex, f' is non-decreasing so

$$\begin{aligned} & f'(x)(x'-x) \leq f(x') - f(x) \leq f'(x')(x'-x) \\ \Leftrightarrow \quad & f(x) \leq f(x') + f'(x)(x-x') \text{ and } f(x') \leq f(x) + f'(x')(x'-x). \end{aligned}$$

Thus, working on $\{f(\bar{x}^k)\}_{k>0}$, the above result tells us that

$$\begin{aligned} f(\bar{x}^{k+1}) &\leq f(\bar{x}^k) + f'(\bar{x}^{k+1})(\bar{x}^{k+1} - \bar{x}^k) \\ &\leq f(\bar{x}^k) + f'(\bar{x}^k)(\bar{x}^{k+1} - \bar{x}^k) + LN(\bar{x}^{k+1} - \bar{x}^k)^2. \end{aligned}$$

where the second inequality comes from the fact that f' is LN-Lipschitz by a straightforward derivation based on Assumption 4.3. Then, using Eq. (4.8), we replace $\bar{x}^{k+1} - \bar{x}^k$ by $-\gamma^k(1/N\mathbb{1}^{\mathrm{T}}\nabla F(x^k))$:

$$\begin{aligned} f(\bar{x}^{k+1}) &\leq f(\bar{x}^k) - \frac{\gamma^k}{N} f'(\bar{x}^k)(\mathbb{1}^{\mathrm{T}}\nabla F(x^k)) + \frac{L(\gamma^k)^2}{N}(\mathbb{1}^{\mathrm{T}}\nabla F(x^k))^2 \\ &\leq f(\bar{x}^k) - \frac{\gamma^k}{N}(f'(\bar{x}^k))^2 + \frac{\gamma^k L}{N}|f'(\bar{x}^k)| \,\|\, \mathbf{J}^{\perp}x^k\|_2 + \frac{L(\gamma^k)^2}{N}(\mathbb{1}^{\mathrm{T}}\nabla F(x^k))^2 \end{aligned}$$

where the inequality is due to the identity $\mathbb{1}^{\mathrm{T}}\nabla F(\bar{x}^k\mathbb{1}) = f'(\bar{x}^k)$ and the fact that $\mathbb{1}^{\mathrm{T}}\nabla F$ is L-Lipschitz continuous from Assumption 4.3. Since the derivatives of the functions $\{f_i\}_{i=1,\dots,N}$ are bounded by Assumption 4.3, we get

$$f(\bar{x}^{k+1}) \leq f(\bar{x}^k) - \frac{\gamma^k}{N}(f'(\bar{x}^k))^2 + C_1\gamma^k \|\, \mathbf{J}^{\perp}x^k\|_2 + C_2(\gamma^k)^2 \tag{4.9}$$

where C_1 and C_2 are finite constants.

Recalling that $\|\, \mathbf{J}^{\perp}x^k\|_2$ is of the same order as γ^k from Eq. (4.6), we see that the two last terms are of the same order as $(\gamma^k)^2$. Thus,

$$f(\bar{x}^{k+1}) \leq f(\bar{x}^k) - \frac{\gamma^k}{N}(f'(\bar{x}^k))^2 + C_3(\gamma^k)^2$$

where C_3 is a finite constant. Iterating this inequality, and using the summability of $(\gamma^k)^2$ (Assumption 4.4), we get that

$$\sum_{k=1}^{\infty} \gamma^k (f'(\bar{x}^k))^2 < \infty.$$

As $\sum_{k>0} \gamma^k = +\infty$, we have $f'(\bar{x}^k) \longrightarrow 0$ as $k \to \infty$.

Conclusion: The fact that $f'(\bar{x}^k) \longrightarrow 0$ implies that $\{\bar{x}^k\}_{k>0}$ does not diverge. Indeed, since f' is non-decreasing and the set $\mathscr{M}$ of its zeros is compact (Assumption 4.3), $\liminf_{|x|\to\infty} |f'(x)| > 0$ which is in contradiction with our result. Thus, the sequence $\{\bar{x}^k\}_{k>0}$ belongs to a compact set, and its accumulation points belong to $\mathscr{M}$. We have shown the following theorem.

Theorem 4.6. *Let Assumptions 4.3, 4.4 and 4.5 hold. Then, the sequence $\{x^k\}_{k>0}$ defined in Eq. (4.4) satisfies $\mathbf{J}^\perp x^k \to 0$ and $\bar{x}^k \to \mathscr{M}$.*

4.2.3 Convergence of the Asynchronous distributed gradient algorithm

As in Section 3.2.2-b, we will now allow the sensors to communicate asynchronously with their neighbors, mathematically this sums up to having the gossip matrix chosen randomly in a set $\mathscr{K}$ through an i.i.d. process as previously described in Section 4.2.1-c. We thus put the following assumptions in place of Assumption 4.5.

Assumption 4.7. *The matrices of $\mathscr{K}$ are non-negative, doubly-stochastic with positive diagonal elements.*

Assumption 4.8. *The gossip matrices sequence $\{\mathbf{K}_{\xi^k}\}_{k>0}$ forms an i.i.d. process valued in $\mathscr{K}$ and $\mathbb{E}[\mathbf{K}]$ is primitive.*

The sketch of the proof is quite similar to the synchronous case up to some modifications in order to take into account the randomness of the gossip matrices. Indeed:

- combining the derivations of Section 3.2.2-b and 4.2.2-a, one can prove that $\mathbb{E}[\|\mathbf{J}^\perp x^k\|_2 |\, \mathscr{F}_{k-1}] = \mathscr{O}(\gamma^k)$ and that $\|\mathbf{J}^\perp x^k\|_2$ converges almost surely to 0;
- remarking that as all the update matrices are column stochastic from Assumption 4.7, Eq. (4.7) holds and thus Eq. (4.9) also holds under Assumption 4.3. Taking the conditional expectation over this inequality and applying Robbins-Siegmund theorem (see [75, Chap 1.3.3] or the original paper [76]) enables to give almost sure convergence results.

Finally, one can find in [71] the following result.

Theorem 4.9. *Let Assumptions 4.3, 4.4, 4.7 and 4.8 hold. Then, the sequence $\{x^k\}_{k>0}$ defined in Eq. (4.2) satisfies $\mathbf{J}^\perp x_k \to 0$ and $\bar{x}^k \to \mathscr{M}$ with probability one.*

4.2.4 Extensions

In the previous sections, we gave assumptions and proofs for the convergence of synchronous and asynchronous distributed gradient algorithms. In particular, we assumed that every agent

could compute exactly the gradient of its cost function at any point. The case where the gradient can only be obtained up to an additional noise leads to the well-studied domain of *distributed stochastic gradient algorithms* [63, 64, 71].

In addition, the considered gossip matrices suffer from the same limitations as in Section 3.2; as they are doubly-stochastic, there is a need for a feedback. Now, if the matrices of $\mathscr{K}$ are i) only row-stochastic; ii) column-stochastic in mean; and iii) $\mathbb{E}[\mathbf{K}]$ is primitive (which was studied in Section 3.2.3 and can represent the *Broadcast Gossip* [7]), the almost sure convergence still holds. Detailed proof and further considerations can be found in [77].

Finally, the algorithms ultimately based on a gradient step followed by a gossipping step (as described in Section 4.2.1-c) can be improved in terms of convergence speed by i) improving the gossip; and/or ii) improving the gradient descent. Unfortunately, improving the convergence speed by improving the average gossip (see Chapter 3) seems to bring only little gains by simulation. Enhancements of the gradient descent through, for example, Nesterov acceleration have been provided in the literature (see [65] and references therein) and showed to offer a significant gain in performances.

Still, first order methods only use local information of the cost functions while the optimum computation can be made more efficient by exploiting the knowledge of the whole functions when available. That is what the next section deals with.

4.3 Distributed Optimization with the ADMM

In this section, we focus on solving Problem 4.2 using a celebrated *proximal algorithm*: the Alternating Direction Method of Multipliers (ADMM). This implies a modification in the network setup: in the previous section the agents were required to compute a gradient whereas now the agents have to evaluate the *proximal operator* of their cost function (denoted by $\mathbf{prox}_f$) which involves a small optimization by itself. After introducing the proximal methods, we will focus on the dual methods of which the ADMM is a variant. Finally, we will explicit the algorithm and see how it applies to our distributed optimization problem.

In the whole section, we will focus on the construction of a distributed optimization algorithm using the ADMM. The reader can refer to [68] and [78] for proofs and further considerations.

4.3.1 Proximal methods

4.3.1-a The proximal operator

Before focusing on the ADMM, let us present the *proximal operator* which is the core of *proximal methods* (see [78] for a detailed overview) to which the ADMM belongs.

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a closed proper convex function, the *proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined as

$$\underset{f}{\mathbf{prox}}(x) = \underset{u}{\operatorname{argmin}} \left\{ f(u) + \frac{1}{2}\|x-u\|_2^2 \right\} \tag{4.10}$$

and is unique for all x under the above assumptions as the function in the argmin is not everywhere infinite and is strongly convex. In most cases, it will be useful to work on the proximal operator of the scaled function $\rho^{-1}f$ with $\rho > 0$ *i.e.*

$$\underset{\rho^{-1}f}{\mathbf{prox}}(x) = \underset{u}{\operatorname{argmin}} \left\{ f(u) + \frac{\rho}{2}\|x-u\|_2^2 \right\}. \tag{4.11}$$

It is interesting in the case of distributed/parallel computing to remark that, for F separable as defined in Problem 4.2, $\mathbf{prox}_F$ is also *separable*:

$$\underset{F}{\mathbf{prox}}(x) = \begin{bmatrix} \mathbf{prox}_{f_1}(x_1) \\ \vdots \\ \mathbf{prox}_{f_N}(x_N) \end{bmatrix}.$$

It is worth to remark that the proximal operator of the *indicator function* of a set C

$$\iota_C(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \end{cases}$$

is simply the (Euclidian) orthogonal projector to C that we denote as Π_C:

$$\underset{\iota_C}{\mathbf{prox}}(x) = \underset{u}{\operatorname{argmin}} \left\{ \iota_C(u) + \frac{1}{2}\|x-u\|_2^2 \right\} = \Pi_C(x). \tag{4.12}$$

4.3.1-b The proximal point algorithm

Now that we have an idea of the behavior of a proximal operator let us focus on the *proximal point algorithm*:

$$x^{k+1} = \underset{\rho^{-1}f}{\mathbf{prox}}(x^k) \tag{4.13}$$

with ρ a positive coefficient. Under suitable assumptions, this algorithm converges to a *fixed point* of $\mathbf{prox}_{\rho^{-1}f}$, that is a point $x^\star$ such that $x^\star = \mathbf{prox}_{\rho^{-1}f}(x^\star)$; recalling Eq. (4.11) it is easy to see that this point minimizes f. Fig. 4.3 illustrates an iteration of the proximal point algorithm with $\rho = 1$ by making appear the quadratic term in the argmin.

Interestingly, when f is differentiable, by differentiating the function inside the argmin and using the fact that x^{k+1} minimizes this function, we have

$$\nabla f(x^{k+1}) + \rho(x^{k+1} - x^k) = 0 \Leftrightarrow x^{k+1} = x^k - \frac{1}{\rho}\nabla f(x^{k+1})$$

so an iteration of the proximal point algorithm can be seen as a gradient descent but the gradient is not taken at the current point but at the arrival point (which is $\mathbf{prox}_{\rho^{-1}f}(x^k)$). That is why proximal algorithms are sometimes called *implicit* gradient methods, as represented by Fig. 4.4.

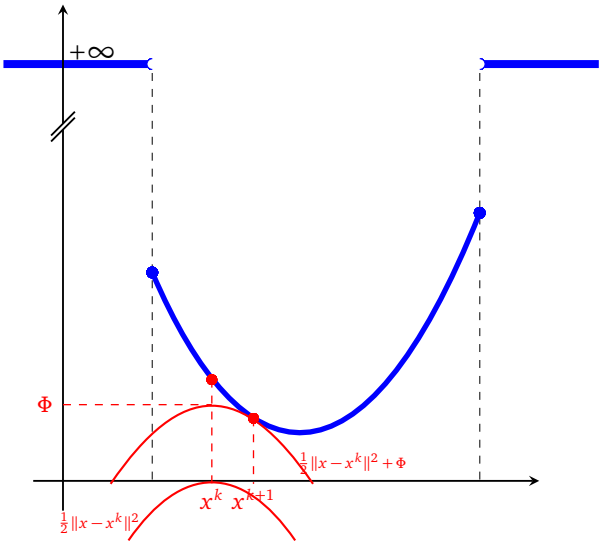


Figure 4.3: Illustration of a proximal iteration on a convex function.

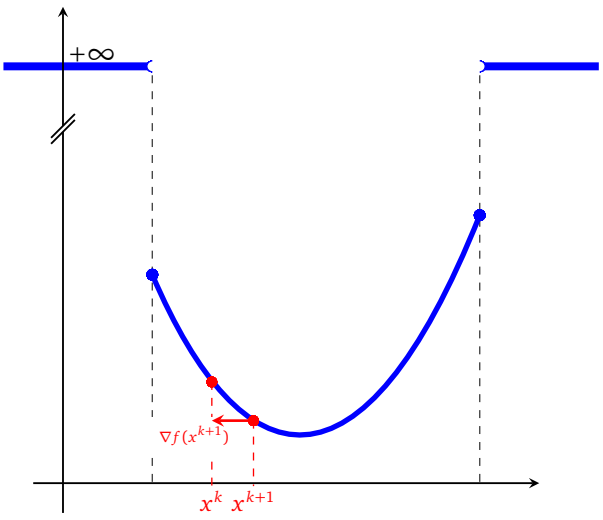


Figure 4.4: Illustration of a proximal iteration with the implicit gradient on a convex function.

An useful property is that, contrary to the gradient algorithm, the iterates $\{x^k\}_{k>0}$ generated by the proximal point algorithm are always *on the same side* of $x^\star$, as

$$\begin{aligned}\langle x^k - x^\star; x^{k+1} - x^\star\rangle &= \langle x^{k+1} - x^\star; x^{k+1} - x^\star\rangle + \langle \frac{1}{\rho}\nabla f(x^{k+1}); x^{k+1} - x^\star\rangle \\ &= \|x^{k+1} - x^\star\|_2^2 + \frac{1}{\rho}\langle \nabla f(x^{k+1}); x^{k+1} - x^\star\rangle \geq 0\end{aligned}$$

where $\langle \nabla f(x^{k+1}); x^{k+1} - x^\star\rangle \geq 0$ as f is convex. By Cauchy-Schwartz inequality, this also implies that $\|x^{k+1} - x^\star\|_2 \leq \|x^k - x^\star\|_2$.

In fact, except in some particular cases, computing a proximal operator is about as hard as minimizing the function itself so the proximal point algorithm has not many direct applications but is often used combined to other optimization techniques. For example, let us consider the problem

Problem 4.10.

$$\min_{x\in\mathbb{R}} \quad f(x)+g(x)$$

where i) f is a smooth convex function, suited for gradient methods; and ii) g is a convex function adapted to proximal operator computation (not necessarily smooth).

A well-known algorithm solving this problem is called *Forward-Backward* splitting where a forward (*explicit*) gradient descent is followed by a backward (*implicit*) gradient, that is to say a proximal step.

Forward-Backward splitting

At each clock tick k:

- Forward step, on f:
$$z^{k+1} = x^k - \gamma^k \nabla f(x^k)$$
- Backward step, on g:
$$x^{k+1} = \underset{\rho^{-1}g}{\mathbf{prox}}(z^{k+1})$$

This algorithm which combines the proximal point algorithm with a gradient is useful in many case, for instance:

- If one wants to minimize f on a constraint convex set C, the gradient algorithm is not adapted as it can result in point outside the constraints set. Taking $g = \iota_C$, one can remark that forward-backward sums up to a projected gradient as the backward step becomes a projection onto C.
- If one wants to find a sparse vector minimize f, it is usual to minimize the sum of f and a penalty function $g(x) = \|x\|_1$ (when f is the Euclidian distance to some vector, this technique is called least absolute shrinkage and selection operator (lasso) [79]). The l_1-norm penalty is non-smooth and thus unsuited to gradient methods (one can use subgradients but the convergence is quite slow).

4.3.2 Constrained problems and Method of Lagrange multipliers

From now on, we will consider constrained problems, that is problems where in addition to minimizing a function, the solution must verify a constraint. We will particularly work on linear equality constraints as they are most commonly used. To treat this kind of optimization problems, it is common to transform the original constrained problem into a new unconstrained problem called the *dual problem*.

4.3.2-a Dual problem

Let us consider a linearly constrained problem as

Problem 4.11.

$$\begin{aligned} \min_{x\in\mathbb{R}^N} \quad & f(x) \\ \textit{subject to} \quad & \boldsymbol{M}x = 0 \end{aligned}$$

with $\mathbf{M} \in \mathbb{R}^{M\times N}$ *the constraint matrix.*

The *Lagrangian* for this problem writes

$$\mathscr{L}(x;\lambda) \triangleq f(x) + \langle \lambda; \mathbf{M}x \rangle \tag{4.14}$$

and the *dual function* is

$$\mathscr{D}(\lambda) \triangleq \inf_x \mathscr{L}(x;\lambda) = -f^*(-\mathbf{M}^{\mathrm{T}}\lambda) \tag{4.15}$$

where λ is named the *dual variable* or *Lagrange multiplier* and f^* denotes the *convex conjugate* (sometimes named *Fenchel-Legendre transformation*) of f (See [72, Chap 3.3] for details about duality).

The *dual problem* of Problem 4.11 (which is called *primal* in opposition) is then

Problem 4.12.

$$\sup_{\lambda\in\mathbb{R}^M} \quad \mathscr{D}(\lambda)$$

where $\mathscr{D}$ *is defined in Eq. (4.15).*

If the respective solutions $x^\star$ and $\lambda^\star$ of Problems 4.11 and 4.12 are such that

$$\sup_\lambda \mathscr{L}(x^\star;\lambda) = \inf_x \mathscr{L}(x;\lambda^\star)$$

we say that $\mathscr{L}$ has a *saddle point*. Then, *strong duality holds* and one can recover a solution of the primal problem from a solution of the dual one if it is unique:

$$x^\star = \operatorname*{argmin}_x \mathscr{L}(x;\lambda^\star).$$

Solving Problem 4.12 using a basic gradient ascent leads to the *dual ascent method*: where $\{\gamma^k\}_{k>0}$ is a positive sequence. We remark that this algorithm involves both the primal and dual variables, it is thus a *primal-dual* algorithm. This algorithm converges under i) strong duality (which imply strict convexity of f); and ii) convergence of the dual ascent (see Section 4.2.1-c) (which imply finiteness of f).

It is interesting to remark that the first step of the algorithm is *separable* if f is. Explicitly, if we replace $f(x)$ by $\sum_{i=1}^N f_i(x_i)$, and define the submatrices $\{\mathbf{M}_i\}_{i=1,\dots,N}$ as $\mathbf{M}x = \sum_{i=1}^N \mathbf{M}_i x_i$, the first step becomes

$$\forall i, \quad x_i^{k+1} = \operatorname*{argmin}_{x_i} f_i(x_i) + \langle \lambda^k; \mathbf{M}_i x_i \rangle$$

which can be computed in parallel. However, the algorithm still needs a fusion center to compute and spread the result of the second step.

Dual ascent

At each clock tick k:

- Primal optimum computation:

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, \mathscr{L}(x; \lambda^k)$$

- Dual ascent:

$$\lambda^{k+1} = \lambda^k + \gamma^k \frac{\partial \mathscr{L}(x^{k+1}; \lambda)}{\partial \lambda} = \lambda^k + \gamma^k (\mathbf{M} x^{k+1})$$

4.3.2-b The Method of Multipliers

In order to bring robustness to the above *dual ascent* algorithm, it is common to consider the *augmented Lagrangian* defined for Problem 4.11 as

$$\mathscr{L}_\rho(x; \lambda) \triangleq f(x) + \langle \lambda; \mathbf{M} x \rangle + \frac{\rho}{2} \|\mathbf{M} x\|_2^2 \tag{4.16}$$

where $\rho > 0$ is a *penalty parameter*. This augmented Lagrangian can be seen as the standard Lagrangian for the following problem which is clearly equivalent to Problem 4.11.

Problem 4.13.

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & f(x) + \tfrac{\rho}{2} \|\mathbf{M} x\|_2^2 \\ \textit{subject to} \quad & \boldsymbol{M} x = 0 \end{aligned}$$

with $\mathbf{M} \in \mathbb{R}^{M \times N}$ *the constraint matrix.*

From this augmented Lagrangian, one can derive an (augmented) dual function and solving our problem using a dual ascent on this function leads to the *Method of Multipliers*.

Method of Multipliers

At each clock tick k:

- Primal optimum computation:

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, \mathscr{L}_\rho(x; \lambda^k)$$

- Dual ascent:

$$\lambda^{k+1} = \lambda^k + \rho (\mathbf{M} x^{k+1})$$

In this algorithm, one must note that the stepsize of the dual gradient ascent is the same as the penalty parameter of the augmented Lagrangian (hence it does not change through time). This method converges under far more general conditions that the simple dual ascent. Unfortunately, even if f is separable, the primal step is not separable anymore in the general case due the quadratic penalty and the matrix $\mathbf{M}$.

4.3.3 Alternating Direction Method of Multipliers

Let us now derive the ADMM which combines the separability of the (simple) dual ascent and the good convergence properties of the method of multipliers. We consider the following split[2] problem and denote by $\mathscr{L}_\rho(x,z;\lambda)$ its augmented Lagrangian.

Problem 4.14.

$$\begin{aligned} \min_{x\in\mathbb{R}^N, z\in\mathbb{R}^M} \quad & f(x)+g(z) \\ \textit{subject to} \quad & \mathbf{M}x = z \end{aligned}$$

with $\mathbf{M}\in\mathbb{R}^{M\times N}$ *the constraint matrix.*

The *ADMM* is a modified version of the method of multipliers where the primal variables are updated in a sequential order[3] instead of being jointly optimized.

Alternating Direction Method of Multipliers

At each clock tick k:

- Alternating primal optimum computation:

$$x^{k+1} = \operatorname*{argmin}_x \mathscr{L}_\rho(x, z^k; \lambda^k)$$

$$z^{k+1} = \operatorname*{argmin}_z \mathscr{L}_\rho(x^{k+1}, z; \lambda^k)$$

- Dual ascent:

$$\lambda^{k+1} = \lambda^k + \rho(\mathbf{M}x^{k+1} - z^{k+1})$$

The iterates $\{x^k\}_{k>0}$ converge to a solution of Problem 4.14 and thus of Problem 4.1 if the function f is closed, proper and convex and if the unaugmented Lagrangian has a saddle point. A convergence proof can be found in [68, Chap. 3.3], but it is quite tedious and not very intuitive. In Section 4.4, we will recall monotone operators that will allow us to give a short and comprehensive proof for the ADMM.

Furthermore, by combining the linear and quadratic terms in the argmin step, one can make the proximal operators of f and g appear.

4.3.4 Distributed optimization using the ADMM

Let us now apply the ADMM to our distributed optimization problem. We will first cast our distributed optimization problem 4.1 into the splitting Problem 4.14 by choosing naively $\mathbf{M} = \mathbf{I}$.

[2] the term *split* comes from the fact that we split the optimization problem into two parts solved by two different variables which must verify a linear constraint.

[3] this is sometimes called a *Gauss-Seidel* pass.

Alternating Direction Method of Multipliers (with explicit argmin **steps)**
At each clock tick k:

- Alternating primal optimum computation:

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f(x) + \frac{\rho}{2} \left\| \mathbf{M}x - z^k + \frac{1}{\rho}\lambda^k \right\|_2^2 \right\} = \underset{\rho^{-1}f \circ \mathbf{M}^{\#}}{\mathbf{prox}} \left(z^k - \frac{1}{\rho}\lambda^k \right)$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \left\{ g(z) + \frac{\rho}{2} \left\| \mathbf{M}x^{k+1} - z + \frac{1}{\rho}\lambda^k \right\|_2^2 \right\} = \underset{\rho^{-1}g}{\mathbf{prox}} \left(\mathbf{M}x^{k+1} + \frac{1}{\rho}\lambda^k \right)$$

- Dual gradient ascent:

$$\lambda^{k+1} = \lambda^k + \rho(\mathbf{M}x^{k+1} - z^{k+1})$$

where $\mathbf{M}^{\#} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}$ is the pseudo-inverse of $\mathbf{M}$.

4.3.4-a A naive algorithm for optimization

Let us rewrite Problem 4.2 as

Problem 4.15.

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^N, z \in \mathbb{R}^N} & F(x) + \iota_{Sp(\mathbb{1})}(z) \\ \textit{subject to} & x = z \end{array}$$

where $F(x) = \sum_{i=1}^{N} f_i(x_i)$ *and* $\iota_{Sp(\mathbb{1})}$ *is the indicator function of the span of* $\mathbb{1}$.

Now, applying the ADMM to this problem, we get the three steps are:

$$\begin{cases} x^{k+1} = \mathbf{prox}_{\rho^{-1}F}\left(z^k - \frac{1}{\rho}\lambda^k\right) \\ z^{k+1} = \mathbf{prox}_{\rho^{-1}\iota_{\mathrm{Sp}(\mathbb{1})}}\left(x^{k+1} + \frac{1}{\rho}\lambda^k\right) = \Pi_{\mathrm{Sp}(\mathbb{1})}\left(x^{k+1} + \frac{1}{\rho}\lambda^k\right) \\ \lambda^{k+1} = \lambda^k + \rho(x^{k+1} - z^{k+1}) \end{cases}$$

where the identity for the second proximal operator comes from Eq. 4.12.

We can remark that i) the first step is separable as F is; and ii) $x^{k+1} + 1/\rho\lambda^k = z^{k+1} + 1/\rho\lambda^{k+1}$ from the third step which implies that $\Pi_{\mathrm{Sp}(\mathbb{1})}\lambda^{k+1} = 0$ by injecting in the second equation, and finally, we have from the third equation again that $\Pi_{\mathrm{Sp}(\mathbb{1})}\lambda^k = 0$ for all $k > 0$. Thus, defining $\{\bar{x}^k\}_{k>0}$ as the sequence of the averages of the $\{x^k\}_{k>0}$, the iterations rewrite more simply by replacing the auxiliary variable z with its value.

$$\begin{cases} x^{k+1} = \mathbf{prox}_{\rho^{-1}F}\left(\bar{x}^k - \frac{1}{\rho}\lambda^k\right) \\ \lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \end{cases}$$

Finally, we see that these two steps can be performed in parallel by each sensor if the average the primal variables are computed and spread by a fusion center at each iteration. This algorithm can be thus be seen as parallel algorithm using a fusion center. Thus, this is unfortunately not a distributed algorithm.

Parallel optimization with the ADMM

At each clock tick k:

- Primal update:

$$\forall i,\ x_i^{k+1} = \underset{\rho^{-1}f_i}{\mathbf{prox}}\left(\bar{x}^k - \frac{1}{\rho}\lambda_i^k\right)$$

- Average computation (by a fusion center):

$$\bar{x}^{k+1} = \frac{1}{N}\mathbb{1}_N^{\mathrm{T}} x^{k+1}$$

- Dual gradient ascent:

$$\forall i,\ \lambda_i^{k+1} = \lambda_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})$$

This algorithm is very interesting even if it is not distributed as one can see that the step that requires a fusion comes directly from the indicator function. Indeed, this function, along with the constraint, enforces the sensors primal variables to be equal with no consideration for the communication graphs and thus the information that the nodes can gather locally. That is why a fusion center is needed when considering Problem 4.15. In Section 4.3.4-b, following the idea of [69], we will design an indicator function and a constraint matrix $\mathbf{M}$ that enforce consensus while only requiring local data gathering.

4.3.4-b A distributed algorithm for optimization

To overcome the need for a global average computation (and thus for a fusion center), [69] proposed to ensure consensus on overlapping subgraphs instead of the whole graph.

From the whole graph $\mathcal{G} = (V, E)$, we construct L subgraphs $\{\mathcal{G}_\ell = (V_\ell, E_\ell)\}_{\ell=1,...,L}$ so that for any subgraph $\mathcal{G}_\ell$, V_ℓ and E_ℓ are a subset of V and E respectively, E_ℓ only links vertices of V_ℓ, and $\mathcal{G}_\ell$ is connected. It is straightforward to see that ensuring the consensus on all the subgraphs imply a consensus on the whole network if

- $\cup_{\ell=1}^{L} V_\ell = V$,
- $(V, \cup_{\ell=}^{L} E_\ell)$ is strongly connected.

Along with every subgraph $\mathcal{G}_\ell$, we define a matrix $\mathbf{M}_\ell \in \mathbb{R}^{|V_\ell| \times N}$ with zeros everywhere except at one entry per line which is 1, the i-th line non null entry corresponds to the number of the node in i-th position in V. Let $M = \sum_{\ell=1}^{L} |V_i|$, we define the matrix $\mathbf{M}$ that will transcribe our partition of the graph as the $\mathbb{R}^{M \times N}$ matrix equal to the column stacking of the submatrices $\{\mathbf{M}_\ell\}_{\ell=1,...,L}$.

It will be useful to introduce the following notations: let z be vector of size M, we split it in blocks of size $|V_1|, ..., |V_L|$ so that $z = (z_{|1}, ..., z_{|L}) \in \mathbb{R}^{|V_1|} \times ... \times \mathbb{R}^{|V_L|}$. This way, if $z = \mathbf{M}x$, we have for all $\ell = 1, ..., L$ $z_{|\ell} = \mathbf{M}_\ell x$. For all $i \in V$ we also define σ_i the set of the indexes of the subgraphs in which i is a node and for all $\ell \in \sigma_i$, we define $z_{i,|\ell}$ as the coefficient of $z_{|l}$ linked to the i-th node.

With these conditions and notations, we formulate the following problem.

Problem 4.16.

$$\begin{aligned} \min_{x\in\mathbb{R}^N, z\in\mathbb{R}^M} \quad & F(x)+G(z) \\ \textit{subject to} \quad & \mathbf{M}x = z \end{aligned}$$

where $F(x) \triangleq \sum_{i=1}^{N} f_i(x_i)$, $G(z) \triangleq \sum_{\ell=1}^{L} \iota_{Sp(\mathbb{1}_{|V_\ell|})}(z_{|\ell})$ *with* $\iota_{Sp(\mathbb{1}_{|V_\ell|})}$ *is the indicator function of the span of* $\mathbb{1}_{|V_\ell|}$.

Applying the ADMM on this problem gives us the following iterations

$$\begin{cases} x^{k+1} = \text{argmin}_x \left\{ F(x) + \frac{\rho}{2} \left\| \mathbf{M}x - z^k + \frac{1}{\rho}\lambda^k \right\|_2^2 \right\} \\ z^{k+1} = \mathbf{prox}_{\rho^{-1}\sum_{\ell=1}^{L} \iota_{\text{Sp}(\mathbb{1}_{|V_\ell|})}(\cdot_\ell)} \left(\mathbf{M}x^{k+1} + \frac{1}{\rho}\lambda^k \right) \\ \lambda^{k+1} = \lambda^k + \rho(\mathbf{M}x^{k+1} - z^{k+1}) \end{cases}$$

where the second step is again a projection Π_S to $S = \text{Sp}(\mathbb{1}_{|V_1|}) \times ... \times \text{Sp}(\mathbb{1}_{|V_L|})$ and combining it with the last equation, we get that for all $k > 0$, $\Pi_S \lambda^k = 0$. The second step is separable and thus rewrites

$$\forall \ell = 1, ..., L \quad z_{|\ell}^{k+1} = \bar{z}_{|\ell}^{k+1} \mathbb{1}_{|V_\ell|} = \frac{1}{|V_\ell|} \sum_{i\in V_\ell} x_i^{k+1} \mathbb{1}_{|V_\ell|}$$

with $\bar{z}_{|\ell}^{k+1}$ the mean of the ℓ-th block of z^{k+1}. This way, by separating the updates of x and λ we get a distributed algorithm.

(Synchronous) Distributed optimization with the ADMM

At each clock tick k:

- Primal update:

$$\forall i,\ x_i^{k+1} = \underset{\rho^{-1} f_i}{\mathbf{prox}} \left(\sum_{\ell\in\sigma_i} \bar{z}_{|\ell}^k - \frac{1}{\rho}\lambda_{i,|\ell}^k \right)$$

- Blockwise average computation :

$$\forall \ell, \bar{z}_{|\ell}^{k+1} = \frac{1}{|V_\ell|} \sum_{i\in V_\ell} x_i^{k+1}$$

- Dual gradient ascent:

$$\forall i, \ell \in \sigma_i,\ \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_{|\ell}^{k+1})$$

This algorithm is distributed at each step: i) in the second step, the average is computed only over the subgraphs whereas it was on the whole graph in the parallel version; ii) the first and third steps are computer locally at each sensor with the knowledge of the averages of the subgraphs in which it is present.

Still, these steps must be performed in the right order which imply that every sensor must have completed its proximal step before computing the subgraph average of next iteration.

This can be costly in terms of convergence time as the proximal operators computational costs can be very different from a sensor to another (*e.g.* if the cost functions rely on datasets of very different sizes). Also, the subgraphs average must be computed at the same time which can result in collisions in networks and thus an increase in the iteration time. This advocates for an asynchronous scheme where a randomly chosen block would perform the three above steps and then pass the token to another block. This will be the topic of a next section after having introduced a useful formalism.

Remark 4.17. *To overcome the synchronism related issues, it was proposed in [70] to update the blocks one by one in a Gauss-Siedel pass instead of altogether. This partially overcome this problem but at the expense of the introduction of a global coordinator.*

4.4 Review on Monotone operators

In this section, we will see that the monotone operators are well suited for describing convex optimization problems and encompasses the ADMM in a simple way.

The results of this section are based on [80, 81] and [82].

4.4.1 Monotone operators

An *operator* T on a Euclidian space[4] $\mathbb{R}^N$ is a set-valued mapping:

$$\begin{array}{rccc} \mathsf{T}: & \mathbb{R}^N & \to & 2^{\mathbb{R}^N} \\ & x & \mapsto & \mathsf{T}(x) \subset \mathbb{R}^N. \end{array}$$

It can equivalently identified to a subset of $\mathbb{R}^N \times \mathbb{R}^N$ and we write $(x,y) \in \mathsf{T}$ when $y \in \mathsf{T}(x)$. We define the *domain* of T as $\mathbf{dom}\,\mathsf{T} = \{x \in \mathbb{R}^N : \mathsf{T}(x) \neq \emptyset\}$ and we will say that T is *single-valued* if for all $x \in \mathbb{R}^N$, $|\mathsf{T}(x)| \leq 1$ (the notation $|\cdot|$ represents here the cardinality of a set). The *identity operator* is I defined as $\{(x,x) : x \in \mathbb{R}^N\}$. We give basic properties for some operators T and U:

- $\forall \rho \in \mathbb{R}$, $\rho\mathsf{T} = \{(x, \rho y) : (x,y) \in \mathsf{T}\}$;
- $\mathsf{T} + \mathsf{U} = \{(x, y+z) : (x,y) \in \mathsf{T}, (x,z) \in \mathsf{U}\}$;
- $\mathsf{T} \circ \mathsf{U} = \{(x,z) : (x,y) \in \mathsf{T}, (y,z) \in \mathsf{U}\}$;
- $\mathsf{T}^{-1} = \{(y,x) : (x,y) \in \mathsf{T}\}$.

An operator T is said to be *monotone* if

$$\forall (x,y), (x',y') \in \mathsf{T}, \quad \langle x - x'; y - y' \rangle \geq 0$$

and such an operator is *maximal* if it is not strictly contained in any other monotone operator as a subset of $\mathbb{R}^N \times \mathbb{R}^N$. We now define two different contraction properties:

[4] this definition and most properties are actually true for any Hilbert space however working in $\mathbb{R}^N$ clarifies the speech.

- T is said to be *non-expansive (NE)* if

$$\forall (x,y),(x',y') \in \mathsf{T}, \quad \|x-x'\| \geq \|y-y'\|$$

- T is said to be *firmly non-expansive (FNE)* if

$$\forall (x,y),(x',y') \in \mathsf{T}, \quad \langle x-x'; y-y' \rangle \geq \|y-y'\|^2$$

where the norm $\|\cdot\|$ is the Euclidian norm. Obviously, a firmly non-expansive operator is non-expansive by Cauchy-Schwartz inequality. Furthermore, both properties imply that the operator is single-valued. Figure 4.5 (which is a reproduction of [81, Fig. 1]) illustrates the above properties by representing the behavior of the vector $y-y'$ with respect to $x-x'$.

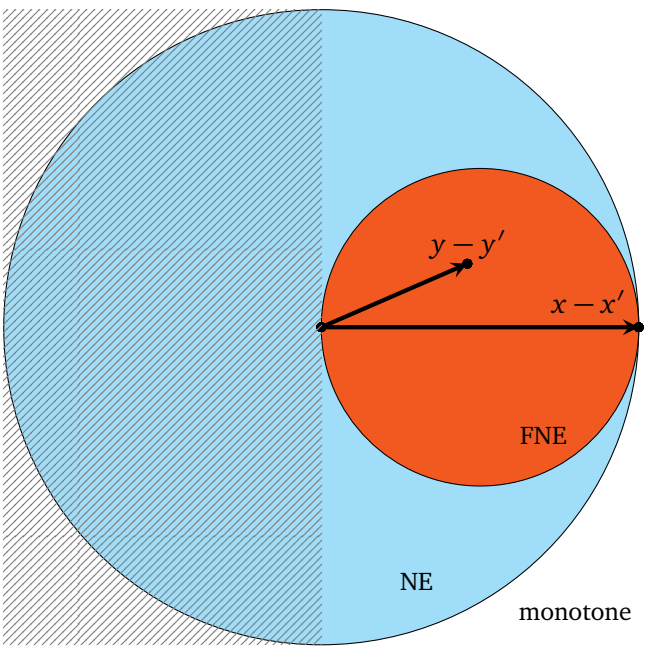


Figure 4.5: Representation of the monotonicity, non-expansiveness and firm non-expansiveness of an operator.

We add the following properties about firmly non-expansive operators:

- T is FNE if and only if $\mathsf{I}-\mathsf{T}$ is FNE;
- T is FNE if and only if $2\mathsf{T}-\mathsf{I}$ is NE;
- T is FNE if and only if $\mathsf{T} = 1/2(\mathsf{C}+\mathsf{I})$ with C NE.

We now define the *zeros* of an operator as

$$\mathbf{zer}\mathsf{T} = \{x : (x,0) \in \mathsf{T}\} = \mathsf{T}^{-1}(0)$$

which will be fundamental in the following.

Remark 4.18. *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a convex function, then its (sub)differential ∂f is a monotone operator and is maximal if f is finite everywhere. Furthermore, finding a zero of ∂f is equivalent to minimizing f.*

4.4.2 The resolvent and the proximal point algorithm

4.4.2-a Definition and properties

We will now see how to find a zero of a monotone operator T. Let us define the *resolvent* of operator T as

$$\mathsf{J}_{\mathsf{T}} \triangleq (\mathsf{I}+\mathsf{T})^{-1}$$

and let us give a fundamental property from [80, 81].

Lemma 4.19. *Let $\rho > 0$. An operator T is monotone if and only if $\mathsf{J}_{\rho\mathsf{T}}$ is firmly non-expansive. Furthermore, T is maximal monotone if and only if $\mathsf{J}_{\rho\mathsf{T}}$ is firmly non-expansive and $\mathbf{dom}\,\mathsf{J}_{\rho\mathsf{T}} = \mathbb{R}^N$.*

Proof. T is monotone if and only if

$$\begin{aligned}
& \forall (x,y),(x',y') \in \mathsf{T}, \ \langle x-x'; y-y' \rangle \geq 0 \\
\Leftrightarrow \ & \forall (x,y),(x',y') \in \mathsf{T}, \ \langle x-x'; \rho y - \rho y' \rangle \geq 0 \\
\Leftrightarrow \ & \forall (x,y),(x',y') \in \mathsf{T}, \ \langle x-x'; x-x'+\rho y - \rho y' \rangle \geq \|x-x'\|^2 \\
\Leftrightarrow \ & \forall (x,x'), \ \langle \mathsf{J}_{\rho\mathsf{T}}(x) - \mathsf{J}_{\rho\mathsf{T}}(x'); x-x' \rangle \geq \|\mathsf{J}_{\rho\mathsf{T}}(x) - \mathsf{J}_{\rho\mathsf{T}}(x')\|^2
\end{aligned}$$

which proves the first part of the theorem. The second part is due to Minty's theorem [83] which states that T is maximal monotone if and only if $\mathbf{im}(\mathsf{I}+\rho\mathsf{T}) = \mathbb{R}^N$, which is equivalent to $\mathbf{dom}(\mathsf{I}+\rho\mathsf{T})^{-1} = \mathbb{R}^N$. □

Another important corollary of the previous result is the so-called *representation lemma*.

Lemma 4.20 (Representation Lemma)**.** *Let $\rho > 0$ and let T be a monotone operator of $\mathbb{R}^N$. Then, every element ζ of $\mathbb{R}^N$ can be written in at most one way as $x + \rho y$ where $(x,y) \in \mathsf{T}$. Furthermore, if T is maximal then every element ζ of $\mathbb{R}^N$ can be written in exactly one way as $x + \rho y$ where $(x,y) \in \mathsf{T}$.*

Proof. If T is monotone, its resolvent is firmly non-expansive and thus single valued so if $\zeta \in \mathbf{dom}\,\mathsf{J}_{\rho\mathsf{T}}$, then there is a unique x such that $\mathsf{J}_{\rho\mathsf{T}}(\zeta) = x \Leftrightarrow \zeta = x + \rho y$ with $(x,y) \in \mathsf{T}$; if $\zeta \notin \mathbf{dom}\,\mathsf{J}_{\rho\mathsf{T}}$ there is no way to write ζ in such a way. If T is also maximal, its resolvent has full-domain and thus the above reasoning is true for all ζ. □

Additionally, for a monotone operator T let us remark that the *fixed points* of J_{T}

$$\mathbf{fix} \ \ \mathsf{J}_{\mathsf{T}} = \{x : (x,x) \in \mathsf{J}_{\mathsf{T}}\}$$

are such that $x = (\mathsf{I}+\mathsf{T})^{-1}(x)$ which is equivalent to $x + \mathsf{T}(x) = x \Rightarrow \mathsf{T}(x) = 0$. So, the fixed points of J_{T} are the zeros of T, we are thus interested in finding fixed points of J_{T}.

Remark 4.21. *From the resolvent* J_T *of an operator* T*, one can define the* Cayley *transform of* T *as*

$$\mathsf{C}_\mathsf{T} \triangleq 2\mathsf{J}_\mathsf{T} - \mathsf{I}.$$

From the properties of the resolvent, one can see that the Cayley transform is non-expansive whenever the resolvent is firmly non-expansive. Furthermore, it is easy to check that the fixed points of the resolvent and the Cayley transform are the same.

4.4.2-b Iterating the resolvent

Finding a fixed point of a function (*i.e.* a single-valued operator) f by fixed point-iterations

$$x^{k+1} = f(x^k) \tag{4.17}$$

is a well-known problem. The most fundamental result is Banach fixed point theorem [84] which states that if f is a *contraction*, that is if there is $0 \leq \alpha < 1$ such that for all x, x', $\alpha\|x - x'\| \geq \|f(x) - f(x')\|$, then f has a unique fixed point $x^\star$ and the sequence $\{x^k\}_{k>0}$ defined by Eq. (4.17) converges to $x^\star$.

Let us now focus on iterating the resolvent of a maximally monotone operator T

$$\zeta^{k+1} = \mathsf{J}_{\rho\mathsf{T}}(\zeta^k) \tag{4.18}$$

with $\rho > 0$. Unfortunately, this resolvent is not a contraction but Lemma 4.19 tells us that it is firmly non-expansive. The two properties are quite different (see Fig. 4.6) as firm non-expansiveness only imply that the operator is 1-Lipschitz (non-expansive) but gives additional monotonicity information. However, one can remark that only one point attain the equality point in the Lipschitz inequality and thus prevents the operator to be a contraction. This point is such that $\zeta - \zeta' = \mathsf{J}_{\rho\mathsf{T}}(\zeta) - \mathsf{J}_{\rho\mathsf{T}}(\zeta')$, taking $\zeta' = \zeta^\star \in \mathbf{fix}\ \mathsf{T}$ implies that $\zeta - \zeta^\star = \mathsf{J}_{\rho\mathsf{T}}(\zeta) - \zeta^\star \Rightarrow \zeta = \mathsf{J}_{\rho\mathsf{T}}(\zeta) \Rightarrow \zeta \in \mathbf{fix}\ \mathsf{T}$. Thus, the only points that do not verify a contraction inequality are the ones we are interested in: the fixed points of T.

The next result will now give a fixed point theorem for the iterations of firmly non-expansive operators due to Rockafellar [80].

Theorem 4.22. *Let* $\rho > 0$*. If* T *is a maximally monotone operator such that* $\mathbf{zer}\ \mathsf{T} \neq \emptyset$*, then the sequence* $\{\zeta^k\}_{k>0}$ *generated by the iteration*

$$\zeta^{k+1} = \mathsf{J}_{\rho\mathsf{T}}(\zeta^k)$$

started at any point converges to a point of $\mathbf{fix}\ \mathsf{J}_{\rho\mathsf{T}}$*.*

Proof. As $\mathbf{zer}\ \mathsf{T} \neq \emptyset$, $\mathbf{fix}\ \mathsf{J}_{\rho\mathsf{T}} \neq \emptyset$ so let us take $\zeta^\star \in \mathbf{fix}\ \mathsf{J}_{\rho\mathsf{T}}$. Then, for all $k > 0$, one has

$$\begin{aligned}
\left\|\zeta^{k+1} - \zeta^\star\right\|^2 &= \left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k) - \zeta^k + \zeta^k - \zeta^\star\right\|^2 \\
&= \left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k) - \zeta^k\right\|^2 + \left\|\zeta^k - \zeta^\star\right\|^2 + 2\left\langle \mathsf{J}_{\rho\mathsf{T}}(\zeta^k) - \zeta^k; \zeta^k - \zeta^\star \right\rangle \\
&= \left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k) - \zeta^k\right\|^2 + \left\|\zeta^k - \zeta^\star\right\|^2 - 2\left\langle \left(\mathsf{I} - \mathsf{J}_{\rho\mathsf{T}}\right)(\zeta^k); \zeta^k - \zeta^\star \right\rangle
\end{aligned}$$

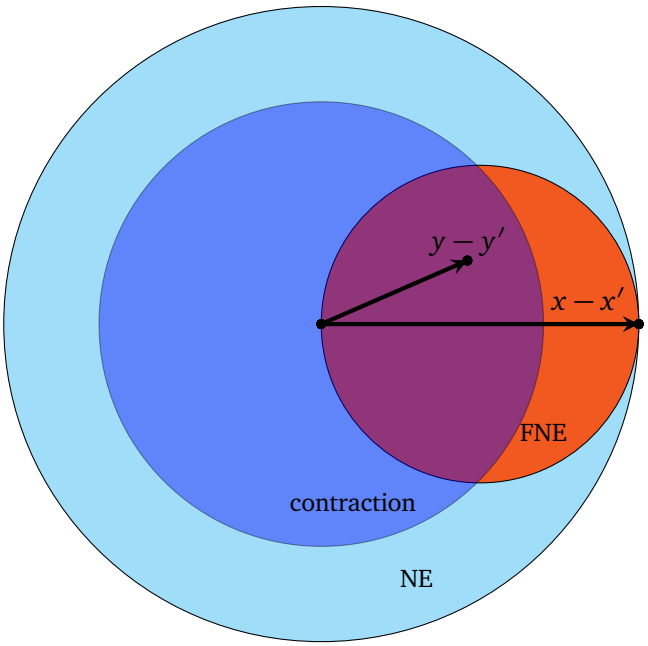


Figure 4.6: Representation of the non-expansiveness, contraction and firm non-expansiveness of an operator.

Now, let us remember that as T is a maximally monotone, $\mathsf{J}_{\rho\mathsf{T}}$ is firmly non-expansive from Lemma 4.19 and so is $\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}}$. Furthermore, as $\mathsf{J}_{\rho\mathsf{T}}(\zeta^\star)=\zeta^\star$, $(\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}})(\zeta^\star)=0$ and so,

$$\begin{aligned}\left\|\zeta^{k+1}-\zeta^\star\right\|^2 &= \left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k)-\zeta^k\right\|^2+\left\|\zeta^k-\zeta^\star\right\|^2\\ &\quad -2\left\langle\left(\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}}\right)(\zeta^k)-\left(\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}}\right)(\zeta^\star);\zeta^k-\zeta^\star\right\rangle\\ &\leq \left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k)-\zeta^k\right\|^2+\left\|\zeta^k-\zeta^\star\right\|^2-2\left\|\left(\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}}\right)(\zeta^k)-\left(\mathsf{I}-\mathsf{J}_{\rho\mathsf{T}}\right)(\zeta^\star)\right\|^2\\ &= \left\|\zeta^k-\zeta^\star\right\|^2-\left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k)-\zeta^k\right\|^2\end{aligned}$$

which implies that $\left\|\zeta^k-\zeta^\star\right\|^2$ converges as it is a non-increasing positive sequence. Also, by iterating we get

$$0\leq\left\|\zeta^0-\zeta^\star\right\|^2-\sum_{k=0}^{\infty}\left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k)-\zeta^k\right\|^2$$

which implies that $\left\|\mathsf{J}_{\rho\mathsf{T}}(\zeta^k)-\zeta^k\right\|^2$ converges to zero. We can thus conclude that i) $\{\zeta^k\}_{k>0}$ is a bounded sequence so it has a least one accumulation point; and ii) any accumulation point ζ^l is such that $\mathsf{J}_{\rho\mathsf{T}}(\zeta^l)=\zeta^l$ *i.e.*$\zeta^l\in$ **fix** $\mathsf{J}_{\rho\mathsf{T}}$. Taking $\zeta^\star=\zeta^l$ an accumulation point, then $\left\|\zeta^{k+1}-\zeta^\star\right\|^2$ goes to zero, so the sequence $\{\zeta^k\}_{k>0}$ converges to a point $\zeta^\star\in$ **fix** $\mathsf{J}_{\rho\mathsf{T}}$. □

4.4.2-c Link to the proximal point algorithm of Section 4.3.1

Now, let us make the connection between the iterations of the resolvent and of the $\mathbf{prox}_f$ operator defined in 4.3.1-a.

For this, we need to put a fundamental assumption on f that we will use extensively in the following. We will say that $f \in \Gamma_0(\mathbb{R}^N)$ if it is a convex, proper and lower semi-continuous (see [82, Def. 1.21]) $\mathbb{R}^N \to \mathbb{R}$ function. This property will enable to state the following result.

Lemma 4.23. *Let $f \in \Gamma_0(\mathbb{R}^N)$ and $\rho > 0$. Then,*

$$\mathsf{J}_{\rho\partial f} = \underset{\rho f}{\mathbf{prox}}.$$

Proof. For all x we have,

$$\begin{aligned}
& y = \mathsf{J}_{\rho\partial f}(x) \\
\Leftrightarrow \quad & y + \rho\partial f(y) = x \\
\Leftrightarrow \quad & \partial f(y) + \frac{1}{\rho}(y - x) = 0 \\
\Leftrightarrow \quad & y = \underset{u}{\operatorname{argmin}} \left\{ f(u) + \frac{1}{\rho}\|u - x\|^2 \right\} \\
\Leftrightarrow \quad & y = \underset{\rho f}{\mathbf{prox}}(x).
\end{aligned}$$

□

Hence, the resolvent iterations of the proximal point algorithm generalizes the proximal operator iterations.

4.4.3 From the proximal point algorithm to the ADMM

The resolvent iterations generalize the proximal operator iterations, thus we should be able to retrieve the above mentioned proximal algorithms and ultimately the ADMM.

4.4.3-a The Method of Multipliers as a proximal point algorithm

Let $f \in \Gamma_0(\mathbb{R}^N)$. We will now consider Problem 4.11 and solve it by solving Problem 4.13. Maximizing $\mathscr{D}$ is the same as minimizing $-\mathscr{D}$ or finding λ such that $\partial\mathscr{D}(\lambda) = -\mathbf{M}\partial f^*(-\mathbf{M}^{\mathrm{T}}\lambda) = 0$ (see the definition of $\mathscr{D}$ in Eq. (4.15)). Using the monotone operators, this sums up to solving the following problem.

Problem 4.24.

$$\textit{Find } \zeta \quad \textit{such that} \quad \mathsf{T}(\zeta) = 0$$

where $\mathsf{T} = -\partial\mathscr{D} = -\mathbf{M}\partial f^* \circ (-\mathbf{M}^{\mathrm{T}})$ is a monotone operator.

We will now see how the proximal point algorithm translates practically. First let us rewrite

$\mathsf{J}_{\rho\mathsf{T}}$ in order to make explicitly appear the subdifferential of f.

$$\begin{aligned}
\mathsf{T} &= -\mathbf{M}\partial f^* \circ (-\mathbf{M}^{\mathrm{T}}) \\
&= \left\{(u,a) : (u,a) \in \mathbb{R}^M \times \mathbb{R}^M, (u,a) \in \mathsf{T}\right\} \\
&= \left\{(u,-\mathbf{M}x) : (u,x) \in \mathbb{R}^M \times \mathbb{R}^N, (-\mathbf{M}^{\mathrm{T}}u, x) \in \partial f^*\right\} \\
&= \left\{(u,-\mathbf{M}x) : (u,x) \in \mathbb{R}^M \times \mathbb{R}^N, (x,-\mathbf{M}^{\mathrm{T}}u) \in \partial f\right\} \\
(\mathsf{I}+\rho\mathsf{T}) &= \left\{(u,u-\rho\mathbf{M}x) : (u,x) \in \mathbb{R}^M \times \mathbb{R}^N, (x,-\mathbf{M}^{\mathrm{T}}u) \in \partial f\right\} \\
\mathsf{J}_{\rho\mathsf{T}} &= \left\{(u-\rho\mathbf{M}x,u) : (u,x) \in \mathbb{R}^M \times \mathbb{R}^N, (x,-\mathbf{M}^{\mathrm{T}}u) \in \partial f\right\} \qquad (4.19)
\end{aligned}$$

where we used the fact that the output of T is necessarily proportional to $-\mathbf{M}$; and the fact that $\partial f^* = (\partial f)^{-1}$ as $f \in \Gamma_0(\mathbb{R}^N)$ (see [82, Cor. 16.24] for details).

The following approach is fundamental as it enable us to derive algorithms from iterations of resolvent. It consists in three main steps: i) the *representation step*; ii) the *mapping step*; and iii) the *re-representation step*. Let us consider iteration k:

$$\zeta^{k+1} = \mathsf{J}_{\rho\mathsf{T}}(\zeta^k)$$

Representation step: we see from above that the input of $\mathsf{J}_{\rho\mathsf{T}}$ writes as $u - \rho\mathbf{M}x$ with $(x,-\mathbf{M}^{\mathrm{T}}u) \in \partial f$. The first step is thus to state that the input, ζ^k, writes uniquely as $u^k - \rho\mathbf{M}x^k$ from the representation lemma (Lemma 4.20):

$$\zeta^k = u^k - \rho\mathbf{M}x^k. \qquad (4.20)$$

Mapping step: from the definition of $\mathsf{J}_{\rho\mathsf{T}}$ and the previous definition of u^k and x^k we see that its output is:

$$\zeta^{k+1} = u^k. \qquad (4.21)$$

Re-representation step: as in the representation step, we need to find the values of u^{k+1} and x^{k+1} with $(x^{k+1},-\mathbf{M}^{\mathrm{T}}u^{k+1}) \in \partial f$, such that ζ^{k+1} writes uniquely as:

$$\zeta^{k+1} = u^{k+1} - \rho\mathbf{M}x^{k+1}. \qquad (4.22)$$

Putting together the three equations above, we get that

$$\begin{aligned}
& u^{k+1} - \rho\mathbf{M}x^{k+1} = u^k \quad \text{with} \quad (x^{k+1},-\mathbf{M}^{\mathrm{T}}u^{k+1}) \in \partial f \\
\Rightarrow\ & u^k + \rho\mathbf{M}x^{k+1} = u^{k+1} \quad \text{with} \quad (x^{k+1},-\mathbf{M}^{\mathrm{T}}u^{k+1}) \in \partial f \\
\Rightarrow\ & -\mathbf{M}^{\mathrm{T}}u^k - \rho\mathbf{M}^{\mathrm{T}}\mathbf{M}x^{k+1} = -\mathbf{M}^{\mathrm{T}}u^{k+1} \in \partial f(x^{k+1}) \\
\Rightarrow\ & 0 \in \partial f(x^{k+1}) + \rho\mathbf{M}^{\mathrm{T}}(\mathbf{M}x^{k+1} + \frac{u^k}{\rho}) \\
\Rightarrow\ & x^{k+1} = \operatorname*{argmin}_x \left\{ f(x) + \frac{\rho}{2}\left\|\mathbf{M}x + \frac{u^k}{\rho}\right\|^2 \right\} = \operatorname*{argmin}_x \mathscr{L}_\rho(x;u^k)
\end{aligned}$$

and the first line also tells us that

$$u^{k+1} = u^k + \rho\mathbf{M}x^{k+1}.$$

So, we get the following couple of iterations,

$$\begin{cases} x^{k+1} = \operatorname{argmin}_x \left\{ f(x) + \frac{\rho}{2} \left\| \mathbf{M}x + \frac{u^k}{\rho} \right\|^2 \right\} = \operatorname{argmin}_x \mathscr{L}_\rho(x; u^k) \\ u^{k+1} = u^k + \rho \mathbf{M} x^{k+1} \end{cases} \tag{4.23}$$

where the first corresponds to a primal variable as it only depends on the primal function minimization and the second corresponds to a dual variable. The obtained algorithm is exactly the Method of multipliers (see Section 4.3.2-b).

Lemma 4.25. *Let $f \in \Gamma_0(\mathbb{R}^N)$ and $\rho > 0$. The proximal point algorithm using the resolvent $\mathsf{J}_{\rho \mathsf{T}}$ of operator $\mathsf{T} \triangleq -\mathbf{M}\partial f^* \circ (-\mathbf{M}^{\mathrm{T}})$ (so that $\mathsf{T} = -\partial \mathscr{D}$) leads to the Method of Multipliers.*

Theorem 4.26. *Let $f \in \Gamma_0(\mathbb{R}^N)$ such that $0 \in \mathbf{core}(\mathbf{M}\,\mathbf{dom}\, f)$ and $\rho > 0$. Then, $\mathsf{T} = -\partial \mathscr{D} = -\mathbf{M}\partial f^* \circ (-\mathbf{M}^{\mathrm{T}})$ is maximal monotone with $\mathbf{zer}\,\mathsf{T} \neq \emptyset$ and the proximal point algorithm*

$$\zeta^{k+1} = \mathsf{J}_{\rho \mathsf{T}}(\zeta^k)$$

converges to a point of $\zeta^\star \in \mathbf{zer}\,\mathsf{T}$ which is dual optimal for Problem 4.11. Furthermore, the intermediate variables $\{x^k\}_{k>0} = \{1/\rho \mathbf{M}^{\#}(\mathsf{J}_{\rho \mathsf{T}} - \mathsf{I})(\zeta^k)\}_{k>0}$ converge to a point $x^\star$ which is primal optimal for Problem 4.11.

Proof. Let $f \in \Gamma_0(\mathbb{R}^N)$, then $f^* \in \Gamma_0(\mathbb{R}^N)$ by [82, Cor. 13.33] and ∂f^* is maximally monotone from [82, Th. 21.2]. This clearly imply that T which is a linear map of a maximally monotone operator is also maximally monotone. If $0 \in \mathbf{core}(\mathbf{M}\,\mathbf{dom}\, f)$ then strong duality holds from Slater condition (see [82, Prop. 26.18] for example) and $\mathbf{zer}\ \mathsf{T} \neq \emptyset$ (see [82, Def. 6.9] for the definition of **core**).

Then, Theorem 4.22 states that $\zeta^{k+1} \to \zeta^\star \in \mathbf{zer}\,\mathsf{T}$ as $k \to \infty$ and $\zeta^\star$ is obviously dual optimal for Problem 4.11. The representation lemma imply that the intermediate sequences $\{x^k\}_{k>0}$ and $\{u^k\}_{k>0}$ both converge and recalling Eq. (4.19), we have the accumulation point is such that $\zeta^\star = u^\star$ and $\mathbf{M}x^\star = 0$. Finally, looking at Eq. (4.23), we see that $x^\star = \operatorname{argmin}_x \mathscr{L}_\rho(x; u^\star)$ and is thus primal optimal thanks to strong duality. □

Putting together Theorem 4.26 and Lemma 4.25 proves the convergence of the Method of Multipliers. The design of the Method of Multipliers in Section 4.3.2-b may seem artificial or tinkered from the Dual Ascent while it is simply a proximal point algorithm applied to the subdifferential of the dual function. In particular, the equality between the penalty parameter of the augmented Lagrangian and the stepsize of the dual ascent is in fact essential.

4.4.3-b Douglas-Rachford splitting and Lions-Mercier operator

We have seen that the Method of Multipliers efficiently solves a minimization problem with a linear equality contraint (Problem 4.11). Applying this method for solving a minimization

problem engaging two functions and a linear equality constraint (Problem 4.14) lead to the following iterations:

$$\begin{cases} (x^{k+1}, z^{k+1}) = \operatorname{argmin}_{(x,z)} \left\{ f(x) + g(z) + \frac{\rho}{2} \left\| \mathbf{M}x - z + \frac{u^k}{\rho} \right\|^2 \right\} = \operatorname{argmin}_{(x,z)} \mathscr{L}_\rho(x, z; u^k) \\ u^{k+1} = u^k + \rho \left(\mathbf{M}x^{k+1} - z^{k+1} \right) \end{cases}$$

but unfortunately, the first step is in general quite hard to compute. Hence, most optimization algorithms solve this problem by *splitting* the argmin between the different functions [85].

In terms of operators, the opposite of the gradient of the dual function writes as

$$-\partial \mathscr{D} = \underbrace{-\mathbf{M} \partial f^* \circ (-\mathbf{M}^{\mathrm{T}})}_{\triangleq \mathsf{T}} + \underbrace{\partial g^*}_{\triangleq \mathsf{U}} \tag{4.24}$$

and, whereas $\mathsf{J}_{\rho(\mathsf{T}+\mathsf{U})}$ is hard to compute (see the above paragraph), $\mathsf{J}_{\rho\mathsf{T}}$ and $\mathsf{J}_{\rho\mathsf{U}}$ are individually easier to compute. A *splitting* proximal algorithm thus employs only the resolvents $\mathsf{J}_{\rho\mathsf{T}}$ and $\mathsf{J}_{\rho\mathsf{U}}$ of T and U and not $\mathsf{J}_{\rho(\mathsf{T}+\mathsf{U})}$.

Many splitting methods exist in the literature (e.g. Peaceman-Rachford) but Douglas-Rachford splitting is the most celebrated one [86]. From this splitting technique, Lions and Mercier [87] derived an operator $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ so that $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ corresponds to Douglas-Rachford splitting of $\mathsf{J}_{\rho(\mathsf{T}+\mathsf{U})}$. This operator writes

$$\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}} = \{(u + \rho b, v - u) : (u, a) \in \mathsf{T}, (v, b) \in \mathsf{U}, u + \rho a = v - \rho b\} \tag{4.25}$$

and thus one has

$$\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} = \mathsf{J}_{\rho\mathsf{T}} \circ \left(2\mathsf{J}_{\rho\mathsf{U}} - \mathsf{I}\right) + \left(\mathsf{I} - \mathsf{J}_{\rho\mathsf{T}}\right) \tag{4.26}$$

or, equivalently

$$\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} = \frac{1}{2}\left(\mathsf{I} + \mathsf{C}_{\rho\mathsf{T}} \circ \mathsf{C}_{\rho\mathsf{U}}\right) \tag{4.27}$$

$$\Leftrightarrow \quad \mathsf{C}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} = \mathsf{C}_{\rho\mathsf{T}} \circ \mathsf{C}_{\rho\mathsf{U}} \tag{4.28}$$

where $\mathsf{C}_{\rho\mathsf{T}}$, $\mathsf{C}_{\rho\mathsf{U}}$, and $\mathsf{C}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ are the *Cayley* transforms of T, U, and $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ respectively (see Remark 4.21).

Let us now give a fundamental result by Lions and Mercier [87] about the monotonicity of this operator.

Theorem 4.27. *Let $\rho > 0$. If T and U are monotone, then $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ is monotone; furthermore, $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ is firmly non-expansive. If T and U are maximal monotone, then $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ is maximal monotone; furthermore, $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ is firmly non-expansive and has full domain.*

Proof. Let $(u, a), (u', a') \in \mathsf{T}$, $(v, b), (v', b') \in \mathsf{U}$ such that $u + \rho a = v - \rho b$ and $u' + \rho a' =$

$v' - \rho b'$. Then,

$$\begin{aligned}
& \left\langle (u' + \rho b') - (u + \rho b); (v' - u') - (v - u) \right\rangle \\
= \; & \rho \left\langle (u' + \rho b') - (u + \rho b); \frac{1}{\rho}(v' - u') - b' - \frac{1}{\rho}(v - u) + b \right\rangle \\
& + \rho \left\langle (u' + \rho b') - (u + \rho b); b' - b \right\rangle \\
= \; & \rho \left\langle u' - u; a' - a \right\rangle + \rho^2 \left\langle b' - b; a' - a \right\rangle \\
& + \rho \left\langle (v' - \rho a') - (v - \rho a); b' - b \right\rangle \\
= \; & \rho \left\langle u' - u; a' - a \right\rangle + \rho^2 \left\langle b' - b; a' - a \right\rangle \\
& + \rho \left\langle v' - v; b' - b \right\rangle - \rho^2 \left\langle a' - a; b' - b \right\rangle \\
= \; & \rho \left\langle u' - u; a' - a \right\rangle + \rho \left\langle v' - v; b' - b \right\rangle
\end{aligned}$$

and so, as long as $\rho > 0$ and both T and U are monotone, $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ is monotone. The firm non-expansiveness of $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ comes directly from Lemma 4.19. Finally, using Minty's theorem [83] and considering Eq. (4.26), one can see that if T and U are maximal monotone, $\mathsf{J}_{\rho\mathsf{T}}$ and $\mathsf{J}_{\rho\mathsf{U}}$ have full domain and so does $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ which concludes the proof. □

This result along with Theorem 4.22 tells us that under standard monotonicity assumptions on T and U, the proximal point algorithm with $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ goes to a fixed point of this resolvent if any. It is thus interesting to characterize the fixed points of $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ *i.e.* the zeros of $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$.

Theorem 4.28. *Let $\rho > 0$. If T and U are monotone then $\zeta \in \mathbf{fix}\, \mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} \iff x \in \mathbf{zer}(\mathsf{T} + \mathsf{U})$ where $x = \mathsf{J}_{\rho\mathsf{U}}(\zeta)$.*

Proof. First, from Eq. (4.27), we get that $\mathbf{fix}\, \mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} = \mathbf{fix}\, \mathsf{C}_{\rho\mathsf{T}} \circ \mathsf{C}_{\rho\mathsf{U}}$. Then, we get from [82, Prop. 25.1(ii)] that $\mathsf{J}_{\rho\mathsf{U}}(\mathbf{fix}\, \mathsf{C}_{\rho\mathsf{T}} \circ \mathsf{C}_{\rho\mathsf{U}}) = \mathbf{zer}(\mathsf{T} + \mathsf{U})$ which (along with the representation lemma) concludes the proof. □

Remark 4.29. *From the two above theorems, we see that if $\rho > 0$ and both T and U are maximal monotone with $\mathbf{zer}(\mathsf{T}+\mathsf{U}) \neq \emptyset$, then $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ is firmly non expansive and thus the induced proximal point algorithm converges to $\zeta^\star$ such that $\mathsf{J}_{\rho\mathsf{U}}(\zeta^\star) \in \mathbf{zer}(\mathsf{T} + \mathsf{U})$. The resolvent of the Lions-Mercier operator has thus the same properties as the resolvent of $\mathsf{T} + \mathsf{U}$ while only depending on the individual resolvents $\mathsf{J}_{\rho\mathsf{T}}$ and $\mathsf{J}_{\rho\mathsf{U}}$.*

4.4.3-c The ADMM as a proximal point algorithm

Let $f, g \in \Gamma_0(\mathbb{R}^N)$. We now solve Problem 4.14 using Douglas-Rachford splitting. For this, we consider the two following operators

$$\mathsf{T} \triangleq -\mathbf{M}\partial f^*(-\mathbf{M}^{\mathrm{T}}) \quad \text{and } \mathsf{U} = \partial g^* \tag{4.29}$$

so that $\mathsf{T}+\mathsf{U} = -\mathscr{D}$ and apply the proximal point algorithm with the resolvent of Lions-Mercier operator. We now write explicitly this resolvent as in Section 4.4.3-a.

$$
\begin{aligned}
\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}} &= \left\{(u+\rho b, v-u) : (u,a) \in -\mathbf{M}\partial f^*(-\mathbf{M}^{\mathrm{T}}), (v,b) \in \partial g^*, u+\rho a = v-\rho b\right\} \\
&= \left\{(u+\rho b, v-u) : (-\mathbf{M}^{\mathrm{T}}u, x) \in \partial f^*, (v,b) \in \partial g^*, u-\rho\mathbf{M}x = v-\rho b\right\} \\
&= \left\{(u+\rho b, v-u) : (x, -\mathbf{M}^{\mathrm{T}}u) \in \partial f, (b,v) \in \partial g, u-\rho\mathbf{M}x = v-\rho b\right\} \\
\mathsf{I}+\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}} &= \left\{(u+\rho b, v+\rho b) : (x, -\mathbf{M}^{\mathrm{T}}u) \in \partial f, (b,v) \in \partial g, u-\rho\mathbf{M}x = v-\rho b\right\} \\
\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}} &= \left\{(v+\rho b, u+\rho b) : (x, -\mathbf{M}^{\mathrm{T}}u) \in \partial f, (b,v) \in \partial g, u-\rho\mathbf{M}x = v-\rho b\right\}
\end{aligned}
$$

We now use the same approach as in Section 4.4.3-a to derive explicit iterations from the proximal point algorithm *i.e.* i) the *representation step*; ii) the *mapping step*; and iii) the *re-representation step*. Let us consider iteration k:

$$\zeta^{k+1} = \mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}(\zeta^k)$$

Representation step: we see from above that the input of $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ writes as $v+\rho b$ with $(v,b) \in \mathsf{U} = \partial g^*$. The first step is thus to state that the input, ζ^k, writes uniquely as $v^k + \rho b^k$ from the representation lemma (Lemma 4.20):

$$\zeta^k = v^k + \rho b^k. \tag{4.30}$$

Mapping step: from the definition of $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ we derive two equalities. First, the equality inside the operator along with the representation lemma and the previous definition of v^k and b^k implies that $v^k - \rho b^k$ writes uniquely[5] as $u - \rho\mathbf{M}x$ with $(u, -\mathbf{M}x) \in \mathsf{T}$:

$$u^{k+1} - \rho\mathbf{M}x^{k+1} = v^k - \rho b^k. \tag{4.31}$$

Secondly, we get that the output of the resolvent is:

$$\zeta^{k+1} = u^{k+1} + \rho b^k. \tag{4.32}$$

Re-representation step: as in the representation step and in order to find the values of v^{k+1} and b^{k+1} with $(v^{k+1}, b^{k+1}) \in \mathsf{U} = \partial g^*$, so that ζ^{k+1} writes uniquely as:

$$\zeta^{k+1} = v^{k+1} + \rho b^{k+1}. \tag{4.33}$$

Now, writing Eq. (4.31) of the mapping step, we get that

$$
\begin{aligned}
& u^{k+1} - \rho\mathbf{M}x^{k+1} = v^k - \rho b^k \quad \text{with} \quad (x^{k+1}, -\mathbf{M}^{\mathrm{T}}u^{k+1}) \in \partial f \\
\Rightarrow \quad & -\rho\mathbf{M}^{\mathrm{T}}\mathbf{M}x^{k+1} - \mathbf{M}^{\mathrm{T}}v^k + \rho\mathbf{M}^{\mathrm{T}}b^k = -\mathbf{M}^{\mathrm{T}}u^{k+1} \in \partial f(x^{k+1}) \\
\Rightarrow \quad & 0 \in \partial f(x^{k+1}) + \rho\mathbf{M}^{\mathrm{T}}\left(\mathbf{M}x^{k+1} - b^k + \frac{v^k}{\rho}\right) \\
\Rightarrow \quad & x^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f(x) + \frac{\rho}{2}\left\|\mathbf{M}x - b^k + \frac{v^k}{\rho}\right\|^2 \right\} = \underset{x}{\operatorname{argmin}}\, \mathscr{L}_\rho(x, b^k; v^k).
\end{aligned}
$$

[5] as they use some properties of the operator, we see them as variables of time $k+1$.

Now combining Eqs. (4.31), (4.32) and (4.33), we have

$$\begin{aligned}
& v^{k+1}+\rho b^{k+1} = u^{k+1}+\rho b^k = v^k+\rho \mathbf{M}x^{k+1} \quad \text{with} \quad (b^{k+1},v^{k+1})\in \partial g \\
\Rightarrow \quad & v^k+\rho\mathbf{M}x^{k+1}-\rho b^{k+1} = v^{k+1}\in \partial g(b^{k+1}) \\
\Rightarrow \quad & 0\in \partial g(b^{k+1})-\rho\left(\mathbf{M}x^{k+1}-b^{k+1}+\frac{v^k}{\rho}\right) \\
\Rightarrow \quad & b^{k+1} = \operatorname*{argmin}_b \left\{ g(b)+\frac{\rho}{2}\left\|\mathbf{M}x^{k+1}-b+\frac{v^k}{\rho}\right\|^2\right\} = \operatorname*{argmin}_b \mathscr{L}_\rho(x^{k+1},b;v^k)
\end{aligned}$$

and the first line also tells us that

$$v^{k+1} = v^k+\rho\left(\mathbf{M}x^{k+1}-b^{k+1}\right).$$

So, we get the following iterations,

$$\left\{\begin{array}{l}
x^{k+1} = \operatorname{argmin}_x \left\{ f(x)+\frac{\rho}{2}\left\|\mathbf{M}x-b^k+\frac{v^k}{\rho}\right\|^2\right\} = \operatorname{argmin}_x \mathscr{L}_\rho(x,b^k;v^k) \\
b^{k+1} = \operatorname{argmin}_b \left\{ g(b)+\frac{\rho}{2}\left\|\mathbf{M}x^{k+1}-b+\frac{v^k}{\rho}\right\|^2\right\} = \operatorname{argmin}_b \mathscr{L}_\rho(x^{k+1},b;v^k) \\
v^{k+1} = v^k+\rho\left(\mathbf{M}x^{k+1}-b^{k+1}\right)
\end{array}\right. \tag{4.34}$$

and as before, x and b are primal variables and v is a dual variable. The obtained algorithm is now exactly the Alternating Direction Method of Multipliers (see Section 4.3.3).

Lemma 4.30. *Let $f,g\in\Gamma_0(\mathbb{R}^N)$ and $\rho>0$. The proximal point algorithm using the resolvent $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ of the Lions-Mercier operator associated with the coefficient ρ and operators $\mathsf{T}\triangleq -\mathbf{M}\partial f^*(-\mathbf{M}^{\mathrm{T}})$ and $\mathsf{U}=\partial g^*$ (so that $\mathsf{T}+\mathsf{U}=-\partial\mathscr{D}$) leads to the Alternating Direction Method of Multipliers.*

Theorem 4.31. *Let $f,g\in\Gamma_0(\mathbb{R}^N)$ such that $0\in\mathbf{core}(\mathbf{dom}\,g-\mathbf{M}\,\mathbf{dom}\,f)$ and $\rho>0$. Then, $\mathsf{T}=-\mathbf{M}\partial f^*\circ(-\mathbf{M}^{\mathrm{T}})$ and $\mathsf{U}=\partial g^*$ are maximal monotone with $\mathbf{zer}(\mathsf{T}+\mathsf{U})\neq\emptyset$ and the proximal point algorithm*

$$\zeta^{k+1} = \mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}(\zeta^k)$$

where $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ is the Lions-Mercier operator (see Eq. (4.25)) converges to a point $\zeta^\star$ such that $v^\star=\mathsf{J}_{\rho\mathsf{U}}(\zeta^\star)\in\mathbf{zer}(\mathsf{T}+\mathsf{U})$ is dual optimal for Problem 4.14. Furthermore, the intermediate variables $\{x^k\}_{k>0}=\{1/\rho\mathbf{M}^{\#}(\mathsf{J}_{\rho\mathsf{T}}-\mathsf{I})(\zeta^k)\}_{k>0}$ converge to a point $x^\star$ which is primal optimal for Problem 4.14.

Proof. Let $f,g\in\Gamma_0(\mathbb{R}^N)$, then $f^*,g^*\in\Gamma_0(\mathbb{R}^N)$ by [82, Cor. 13.33] and ∂f^* , ∂g^* are maximally monotone from [82, Th. 21.2]. This clearly implies that T and U are maximal monotone. Let us consider Problem 4.14 (which is equivalent to the problem of [82, Def. 15.19]); if $0\in\mathbf{core}(\mathbf{dom}\,g-\mathbf{M}\,\mathbf{dom}\,f)$, strong duality holds from [82, Prop. 15.22] and $\mathbf{zer}(\mathsf{T}+\mathsf{U})\neq\emptyset$.

Then, Theorem 4.22 states that $\zeta^{k+1}\to\zeta^\star\in\mathbf{zer}\,\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}=$ as $k\to\infty$ and Theorem 4.28 tells us that $v^\star=\mathsf{J}_{\rho\mathsf{U}}(\zeta^\star)\in\mathbf{zer}(\mathsf{T}+\mathsf{U})$ is dual optimal. Using the fact that $v^\star$ is a zero of $\mathsf{T}+\mathsf{U}$, we get that $\mathsf{T}+\mathsf{U}\ni(v^\star,-\mathbf{M}x^\star+b^\star)=(v^\star,0)$ so $\mathbf{M}x^\star=b^\star$ which implies that the found zero is feasible. Finally, looking at Eq. (4.34), we see that $x^\star=\operatorname{argmin}_x\mathscr{L}_\rho(x,b^\star;u^\star)$ and is thus primal optimal thanks to strong duality. □

Putting together Theorem 4.31 and Lemma 4.30 prove the convergence of the Alternating Direction Method of Multipliers. The formalism of monotone operators enabled us to derive easily proximal algorithms and give precise and flexible proofs for their convergence. In the next section, we will build a asynchronous optimization algorithm based on the ADMM from a randomized proximal point algorithm.

4.5 Asynchronous Distributed Optimization using random ADMM

In this section we develop our main results about the design and convergence of an asynchronous distributed optimization algorithm based on the ADMM. For this purpose, we prove a general result about the convergence of a random version of the proximal point algorithm.

4.5.1 Motivation

Recalling Section 4.3.4-b, an efficient technique to perform distributed optimization on a network is to partition the underlying graph into subgraphs and then solve Problem 4.16 which is equivalent to Problem 4.2 under some conditions about the subgraphs. We saw that using standard ADMM to solve this problem leads to a distributed algorithm where the computations are done locally and the communications only involve neighbors. However, the three steps must be performed in the right order, one after another which is unfortunate as the proximal step which is performed individually at each node may take very different computation times between the sensors; for example, in the case of learning with heterogeneous datasets. Hence, it would be pertinent to update only one subgraph at a time randomly.

Looking at the distributed algorithm, it is clear that the M-sized variables (namely z and λ) are updated block by block, where a block represent a subgraph. Thus, recalling the previous section, updating only one block seems similar to operating a Lions-Mercier resolvent where only the components of the block are kept while the others stay the same.

We will first formalize the notion of block operators and prove a new convergence result about a block-random proximal point algorithm. Then, we will derive a new asynchronous optimization algorithm based on block-random iterations of the resolvent of Lions-Mercier operator.

4.5.2 Subgraphs and block-variables

As in Section 4.3.4-b, we divide the underlying graph $\mathcal{G} = (V, E)$ into L subgraphs $\{\mathcal{G}_\ell = (V_\ell, E_\ell)\}_{\ell=1,\dots,L}$. For any variable ζ of size $M \triangleq \sum_{\ell=1}^{L} |V_\ell|$, we define the ℓ-th *block-variable* (of size $|V_\ell|$) by $\zeta_{|\ell}$ so that

$$\zeta = \begin{pmatrix} \zeta_{|1} \\ \vdots \\ \zeta_{|L} \end{pmatrix} \quad \text{where } \forall \ell, \;\; \zeta_{|\ell} = \left(\zeta_{i,|\ell}\right)_{i \in V_\ell}.$$

The matrix linking the nodes of the graph V to the subgraphs nodes $V_1 \times ... \times V_L$ is denoted by $\mathbf{M} \in \mathbb{R}^{M \times N}$. As with the variables, we divide this matrix into L blocks so that

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_L \end{pmatrix} \quad \text{where } \forall \ell, \ \ \mathbf{M}_\ell \in \mathbb{R}^{|V_\ell| \times N}.$$

where each $\mathbf{M}_\ell$ has only one non-null coefficient per line which is 1 so that if $V_\ell = (i_1, ..., i_{|V_\ell|}) \in V^{|V_\ell|}$, $(\mathbf{M}_\ell)_{1,i_1} = 1$ and so on... For any node $i \in V$, we define the set of blocks (*i.e.*subgraphs) to which it belongs by $\sigma_i = \{\ell \in \{1, ..., L\} : i \in V_\ell\}$. With this notation, if $z = \mathbf{M}x$, we have that for all $i \in V$ and all $\ell \in \sigma_i$ that $z_{i,|\ell} = x_i$.

In terms of problem reformulation, we replace the consensus indicator $\iota_{\mathrm{Sp}(\mathbb{1}_N)}(z)$ of Problem 4.15 with $G(z) \triangleq \sum_{\ell=1}^{L} \iota_{\mathrm{Sp}(\mathbb{1}_{|V_\ell|})}(z_{|\ell})$ where $\iota_{\mathrm{Sp}(\mathbb{1}_{|V_\ell|})}$ is the indicator function of the span of $\mathbb{1}_{|V_\ell|}$ to obtain Problem 4.16. The two problems are equivalent under the following assumption.

Assumption 4.32. *In order to have* $G(z) \triangleq \sum_{\ell=1}^{L} \iota_{Sp(\mathbb{1}_{|V_\ell|})}(z_{|\ell}) = \iota_{Sp(\mathbb{1}_M)}(z)$*, we assume that*

- $\cup_{\ell=1}^{L} V_\ell = V$*;*
- $(V, \cup_{\ell=}^{L} E_\ell)$ *is strongly connected.*

This way, we obtained a problem fully separable between the blocks of the variables. Now, let us see how the proximal point algorithm behaves when it is updated block per block randomly.

4.5.3 Random Gauss-Seidel iterations on the proximal point algorithm

From a single valued operator T of $\mathbb{R}^M$ we define for all $\ell = 1, .., L$ the *block-mapping* $\check{\mathsf{T}}^{|\ell} : \mathbb{R}^M \to \mathbb{R}^{|V_\ell|}$ so that for all $\zeta \in \mathbb{R}^M$,

$$\mathsf{T}(\zeta) = \begin{pmatrix} \check{\mathsf{T}}^{|1}(\zeta) \\ \vdots \\ \check{\mathsf{T}}^{|L}(\zeta) \end{pmatrix}$$

and additionally we define for all $\ell = 1, .., L$ the *block-operator* $\mathsf{T}^{|\ell} : \mathbb{R}^M \to \mathbb{R}^M$ verifying

$$\mathsf{T}^{|\ell}(\zeta) = \begin{pmatrix} \zeta_{|1} \\ \vdots \\ \check{\mathsf{T}}^{|\ell}(\zeta) \\ \vdots \\ \zeta_{|L} \end{pmatrix}.$$

Let T be a firmly non-expansive operator, and considering the proximal point algorithm iterations

$$\zeta^{k+1} = \mathsf{T}(\zeta^k)$$

its block *Gauss-Seidel* version then writes

$$\zeta^{k+1} = \mathsf{T}^{|L} \circ \cdots \circ \mathsf{T}^{|1}(\zeta^k).$$

As mentioned before, we are interested in a randomized version of this block Gauss-Seidel pass; namely, let us define the block-selection process $\{\xi^k\}_{k>0}$ and let us put the following assumption.

Assumption 4.33. *The process $\{\xi^k\}_{k>0}$ is independent and identically distributed on $\{1,...,L\}$ with $\mathbb{P}[\xi^1 = \ell] = p_\ell > 0$ for all $\ell = 1,...,L$.*

We now state our main contribution.

Theorem 4.34. *Let Assumption 4.33 hold. Let T be a firmly non-expansive operator with full domain such that $\mathbf{fix}\,\mathsf{T} \neq \emptyset$. Then, for any initial value, the sequence $\{\zeta^k\}_{k>0}$ produced by iterations*

$$\zeta^{k+1} = \mathsf{T}^{|\xi^{k+1}}(\zeta^k)$$

converges almost surely to a point of $\mathbf{fix}\,\mathsf{T}$.

Proof. Let us define the diagonal matrix $\mathbf{P} \in \mathbb{R}^{M\times M}$ such that the entries of the ℓ-th block are $p_\ell^{-1/2}$. We denote by $\|\cdot\|_{\mathbf{P}}$ the vector norm weighted by matrix $\mathbf{P}$ so that $\|\zeta\|_{\mathbf{P}}^2 = \sum_{\ell=1}^L p_\ell^{-1}\|\zeta_{|\ell}\|^2$. Fix $\zeta^\star \in \mathbf{fix}\,\mathsf{T}$. Conditionally to the sigma-field of the past selections $\mathscr{F}_k = \sigma(\xi^1,...,\xi^k)$, we get

$$\begin{aligned}
\mathbb{E}\Big[\, \big\|\zeta^{k+1}-\zeta^\star\big\|_{\mathbf{P}}^2 \Big| \mathscr{F}_k \,\Big] &= \sum_{\ell=1}^L p_\ell \big\|\mathsf{T}^{|\ell}(\zeta^k)-\zeta^\star\big\|_{\mathbf{P}}^2 \\
&= \sum_{\ell=1}^L p_\ell \left(\frac{1}{p_\ell}\left\|\mathring{\mathsf{T}}^{|\ell}(\zeta^k)-\zeta^\star_{|\ell}\right\|^2 + \sum_{\substack{\ell'=1\\ \ell'\neq\ell}}^L \frac{1}{p_{\ell'}}\left\|\zeta^k_{\ell'}-\zeta^\star_{|\ell'}\right\|^2 \right) \\
&= \big\|\mathsf{T}(\zeta^k)-\zeta^\star\big\|^2 + \sum_{\ell=1}^L \frac{1-p_\ell}{p_\ell}\left\|\zeta^k_{|\ell}-\zeta^\star_{|\ell}\right\|^2 \\
&= \big\|\zeta^k-\zeta^\star\big\|_{\mathbf{P}}^2 + \big\|\mathsf{T}(\zeta^k)-\zeta^\star\big\|^2 - \big\|\zeta^k-\zeta^\star\big\|^2
\end{aligned}$$

Now, using the fact that for a firmly non-expansive operator (see the proof of Theorem 4.22),

$$\big\|\mathsf{T}(\zeta^k)-\zeta^\star\big\|^2 - \big\|\zeta^k-\zeta^\star\big\|^2 \leq -\big\|\mathsf{T}(\zeta^k)-\zeta^k\big\|^2$$

we get a fundamental inequality

$$\mathbb{E}\Big[\, \big\|\zeta^{k+1}-\zeta^\star\big\|_{\mathbf{P}}^2 \Big| \mathscr{F}_k \,\Big] \leq \big\|\zeta^k-\zeta^\star\big\|_{\mathbf{P}}^2 - \big\|\mathsf{T}(\zeta^k)-\zeta^k\big\|^2 \tag{4.35}$$

which shows that $\{\|\zeta^k-\zeta^\star\|_{\mathbf{P}}^2\}_{k>0}$ is a non-negative super-martingale with respect to the filtration $\{\mathscr{F}_k\}_{k>0}$. Hence, it converges to a non-negative random variable $X_{\zeta^\star}$ with probability one.

Given a countable dense subset $\mathbb{F}$ of $\mathbf{fix}\,\mathsf{T}$, we have that $\|\zeta^k-\zeta\|_{\mathbf{P}}\to X_\zeta$ for all $\zeta\in\mathbb{F}$ with probability one. Let $\zeta^\star\in\mathbf{fix}\,\mathsf{T}$ and let $\varepsilon>0$, as $\mathbb{F}$ is dense one can choose $\zeta\in\mathbb{F}$ such that $\|\zeta-\zeta^\star\|_{\mathbf{P}}<\varepsilon$. Putting together these two assertion, we get that with probability one,

$$\|\zeta^k-\zeta^\star\|_{\mathbf{P}}\leq\|\zeta^k-\zeta\|_{\mathbf{P}}+\|\zeta-\zeta^\star\|_{\mathbf{P}}\leq X_\zeta+2\varepsilon$$

for k large enough. Similarly, one has

$$\|\zeta^k-\zeta^\star\|_{\mathbf{P}}\geq\|\zeta^k-\zeta\|_{\mathbf{P}}-\|\zeta-\zeta^\star\|_{\mathbf{P}}\geq X_\zeta-2\varepsilon$$

which implies that

C1: There is a probability one set onto which $\|\zeta^k-\zeta^\star\|_{\mathbf{P}}$ converges for all $\zeta^\star\in\mathbf{fix}\,\mathsf{T}$.

Taking the expectation onto Eq. (4.35) and iterating over k, we get

$$\sum_{k=0}^{\infty}\mathbb{E}\left[\left\|\mathsf{T}(\zeta^k)-\zeta^k\right\|^2\right]\leq\left\|\zeta^0-\zeta^\star\right\|_{\mathbf{P}}^2 \tag{4.36}$$

which implies by Markov's inequality and Borel-Cantelli lemma that

C2: $\left\|\mathsf{T}(\zeta^k)-\zeta^k\right\|^2\to 0$ almost surely.

We now consider an elementary event in the probability one set where **C1** and **C2** hold. On this event, **C1** implies that $\{\xi^k\}_{k>0}$ is bounded and thus has accumulation points from Bolzano-Weierstrass theorem; **C2** imply that the accumulation points are in $\mathbf{fix}\,\mathsf{T}$. Assume that $\zeta_a^\star\in\mathbf{fix}\,\mathsf{T}$ is an accumulation point, then $\|\zeta^k-\zeta_a^\star\|_{\mathbf{P}}$ converges and $\lim\|\zeta^k-\zeta_a^\star\|_{\mathbf{P}}=\liminf\|\zeta^k-\zeta_a^\star\|_{\mathbf{P}}=0$ which shows that $\xi_a^\star$ is unique. □

We just proved that the proximal point algorithm can be performed randomly per block without loss in the convergence properties.

4.5.4 Asynchronous Distributed Optimization with the ADMM

Let $f,g\in\Gamma_0(\mathbb{R}^{\mathbb{N}})$. Recalling the computations of Section 4.4.3, the block random iterations of the proximal point algorithm with the resolvent of Lions-Mercier operator writes

$$\zeta^{k+1}=\mathsf{J}^{|\xi^{k+1}}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}(\zeta^k)$$

which is the same as

$$\begin{cases}\tilde{\zeta}^{k+1}=\mathsf{J}^{|\xi^{k+1}}_{\mathsf{S}_{\rho,\mathsf{A},\mathsf{B}}}(\zeta^k)\\ \zeta^{k+1}_{|\xi^{k+1}}=\tilde{\zeta}^{k+1}_{|\xi^{k+1}}\quad\text{and }\forall\ell\neq\xi^{k+1},\ \zeta^{k+1}_{|\ell}=\zeta^k_{|\ell}\end{cases}$$

so at time k the update of the ξ^{k+1}-th block of ζ is the same as in Section 4.4.3 whereas the others keep their previous value.

Following the proof of Theorem 4.31 except for the convergence of the proximal point algorithm which is now provided by Theorem 4.34, we get the following theorem.

Theorem 4.35. *Let $f, g \in \Gamma_0(\mathbb{R}^N)$ such that $0 \in \mathbf{core}(\mathbf{dom}\, g - \mathbf{M}\,\mathbf{dom}\, f)$ and $\rho > 0$. Let Assumptions 4.32 and 4.33 hold. Then, $\mathsf{T} = -\mathbf{M}\partial f^* \circ (-\mathbf{M}^{\mathrm{T}})$ and $\mathsf{U} = \partial g^*$ are maximal monotone with $\mathbf{zer}(\mathsf{T}+\mathsf{U}) \neq \emptyset$ and the randomized Gauss-Seidel proximal point algorithm*

$$\zeta^{k+1} = \mathsf{J}^{|\xi^{k+1}}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}(\zeta^k)$$

where $\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}$ is the Lions-Mercier operator (see Eq. (4.25)) converges almost surely to a point $\zeta^\star$ such that $v^\star = \mathsf{J}_{\rho\mathsf{U}}(\zeta^\star) \in \mathbf{zer}(\mathsf{T}+\mathsf{U})$ is dual optimal for Problem 4.14. Furthermore, the intermediate variables $\{x^k\}_{k>0} = \{1/\rho \mathbf{M}^{\#}(\mathsf{J}_{\rho\mathsf{T}} - \mathsf{I})(\zeta^k)\}_{k>0}$ converge almost surely to a point $x^\star$ which is primal optimal for Problem 4.14.

Derivation of our asynchronous optimization algorithm based on the ADMM: using the representation steps of Eqs. (4.30) and (4.33), one can see that only the ξ^{k+1}-th block of variables v and b need to be updated. Then, looking at Eq. (4.34) we see that these update need implicitly the update of the variable x; more precisely, only the update of the ξ^{k+1}-th block of $\mathbf{M}x$ is needed, which corresponds to the $\{x_i\}_{i\in V_{\xi^{k+1}}}$. The iterations of this new algorithm thus write

$$\begin{cases} \mathbf{M}_{\xi^{k+1}} x = \operatorname{argmin}_x \left\{ f_{|\xi^{k+1}}(x) + \frac{\rho}{2} \left\| \mathbf{M}_{\xi^{k+1}} x - b^k_{|\xi^{k+1}} + \frac{v^k_{|\xi^{k+1}}}{\rho} \right\|^2 \right\} \\ b^{k+1}_{|\xi^{k+1}} = \operatorname{argmin}_b \left\{ g_{|\xi^{k+1}}(b) + \frac{\rho}{2} \left\| \mathbf{M}_{\xi^{k+1}} x^{k+1} - b + \frac{v^k_{|\xi^{k+1}}}{\rho} \right\|^2 \right\} \\ v^{k+1}_{|\xi^{k+1}} = v^k_{|\xi^{k+1}} + \rho \left(\mathbf{M}_{\xi^{k+1}} x^{k+1} - b^{k+1}_{|\xi^{k+1}} \right) \end{cases} \tag{4.37}$$

where $f_{|\xi^{k+1}}$ and $g_{|\xi^{k+1}}$ are the functions f and g restricted in their operands to the ξ^{k+1}-th block. The entries that are not linked to the current block stay the same. Finally, combining these randomized iterations of the ADMM with the separable functions defined in Problem 4.16, we are able to derive a fully-distributed optimization algorithm based on the ADMM.

Lemma 4.36. *Let $f \in \Gamma_0(\mathbb{R}^N)$ be separable, $g \in \Gamma_0(\mathbb{R}^M)$ be defined as in Problem 4.16, and $\rho > 0$. The randomized Gauss-Seidel proximal point algorithm of Theorem 4.34 using the resolvent $\mathsf{J}_{\mathsf{S}_{\rho,\mathsf{T},\mathsf{U}}}$ of the Lions-Mercier operator associated with the coefficient ρ and operators $\mathsf{T} \triangleq -\mathbf{M}\partial f^*(-\mathbf{M}^{\mathrm{T}})$ and $\mathsf{U} = \partial g^*$ (so that $\mathsf{T}+\mathsf{U} = -\partial \mathscr{D}$) leads to the Asynchronous distributed optimization with the Alternating Direction Method of Multipliers.*

Hence, our proposed algorithm converges using Theorem 4.35.

We can see that this algorithm is well suited for a fully-distributed implementation as at each time only one subgraph needs to compute and exchange information while the other stay completely idle. Each sensor i has to know: i) its own function f_i, the common parameter ρ and how to perform a proximal operator; and ii) the variables $z_{|\ell}$ and $\lambda_{|\ell}$ for all $\ell \in \sigma_i$ and at each time (this is not limiting as, at each iteration, a new version of these variables are computed, it must take part to the computation). Then, the nodes only exchange their values locally in the subgraphs without any fusion center.

Asynchronous distributed optimization with the ADMM

At each clock tick k, let ξ^{k+1} be the index of the activating subgraph:

- Primal update in the block:

$$\begin{aligned}\forall i \in V_{|\xi^{k+1}},\ x_i^{k+1} &= \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho\sigma_i}{2} \left\| x - \frac{1}{\sigma_i}\left(\sum_{\ell\in\sigma_i} \bar{z}_{|\ell}^k - \frac{1}{\rho}\lambda_{i,|\ell}^{k+1} \right) \right\|^2 \right\} \\ &= \underset{\sigma_i^{-1}\rho^{-1} f_i}{\mathbf{prox}} \left(\frac{1}{\sigma_i} \sum_{\ell\in\sigma_i} \bar{z}_{|\ell}^k - \frac{1}{\rho}\lambda_{i,|\ell}^{k+1} \right)\end{aligned}$$

- Block average computation :

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|V_{|\xi^{k+1}}|} \sum_{i\in V_{|\xi^{k+1}}} x_i^{k+1}$$

- Dual ascent in the subgraph:

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^{k} + \rho(x_i^{k+1} - \bar{z}_{|\xi^{k+1}}^{k+1})$$

- The other blocks do not change their values:

$$\forall i \notin V_{\xi^{k+1}},\ \ x_i^{k+1} = x_i^k \qquad \forall \ell \neq \xi^{k+1},\ \ z_{|\ell}^{k+1} = z_{|\ell}^k, \lambda_{|\ell}^{k+1} = \lambda_{|\ell}^k$$

Remark 4.37. *Taking only one subgraph equal to the original graph gives the original Distributed optimization with the ADMM. Furthermore, if we choose as subgraphs each edges of the original graph with the two ends ($\mathcal{G}_e = ((i,j),\{i,j\})$, $e = \{i,j\} \in E$), then the block average computation is similar to the* Random Gossip *algorithm (see Section 3.2.2-b).*

4.6 Numerical Illustrations

Let us consider the 5-agents network depicted in Fig. 4.7 and divide it into subgraphs so that each subgraph contains one edge of the original graph and thus two nodes; this gives us 5 subgraphs as seen in Fig. 4.8. All simulations presented in this section will use these graph and subdivisions.

We choose to illustrate the performance of the algorithms mentioned in this chapter using quadratic functions as represented in Fig. 4.9. This comes from two main reasons: i) for a fair use of the gradient, the functions have to be smooth; and ii) quadratic (norms) functions are often used in estimation (without a penalty term) and thus can be met in practice.

In Fig. 4.10, we plot the Squared Error versus the number of iterations for a realization of various distributed optimization algorithm. We compared two first-order algorithms and two ADMM-based ones; for each couple, we took a synchronous and an asynchronous algo-

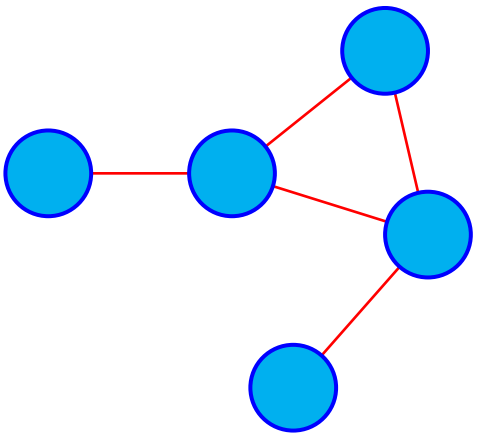
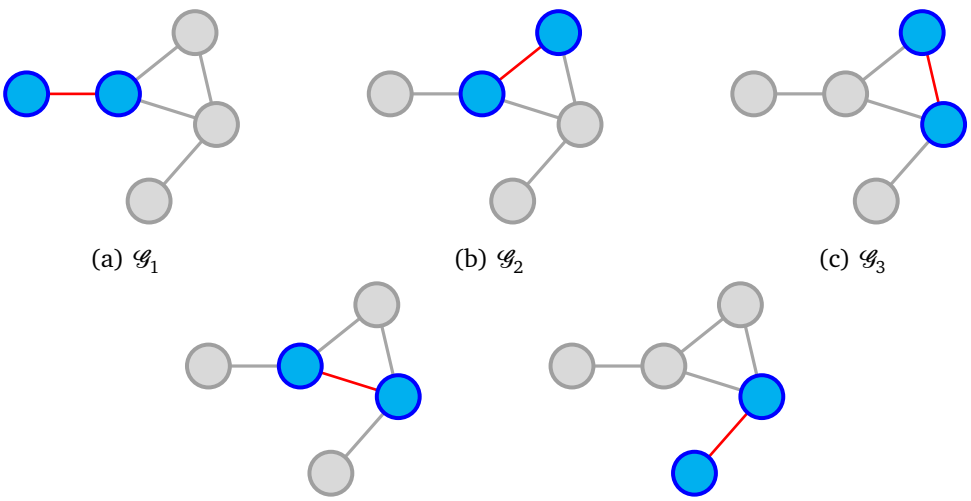


Figure 4.7: Considered underlying network.



(a) $\mathcal{G}_1$ (b) $\mathcal{G}_2$ (c) $\mathcal{G}_3$

(d) $\mathcal{G}_4$ (e) $\mathcal{G}_5$

Figure 4.8: Subgraphs obtained by pairwise division.

rithm. Namely, we plot i) the *Synchronous Distributed gradient descent* with $\gamma^k = 1/k$ (see Section 4.2.2); ii) the *Asynchronous Distributed gradient descent* with $\gamma^k = 1/k$ (see Section 4.2.3); iii) the *Synchronous Distributed optimization with the ADMM* with $\rho = 1$ (see Section 4.3.4-b); and iv) the proposed *Asynchronous Distributed optimization with the ADMM* with $\rho = 1$ (see Section 4.5.4). As expected, the gradient-based algorithms both converge rather slowly compared to the ADMM-based ones whose convergence is clearly exponential. It is very interesting to remark that the asynchronous gradient fluctuates quite a lot which prevents to use a stopping criterion based on the iterations convergence (for instance, in practical systems it is often interesting to stop the algorithm when $\|x^{k+1} - x^k\| \leq \varepsilon$ for some $\varepsilon > 0$ in order not to waste computational time to acquire an unnecessary precision). Finally, we remark that even if less quick, our proposed asynchronous ADMM has the same linear convergence as the synchronous distributed ADMM; furthermore, it does not fluctuate a lot while decreasing, thus, it

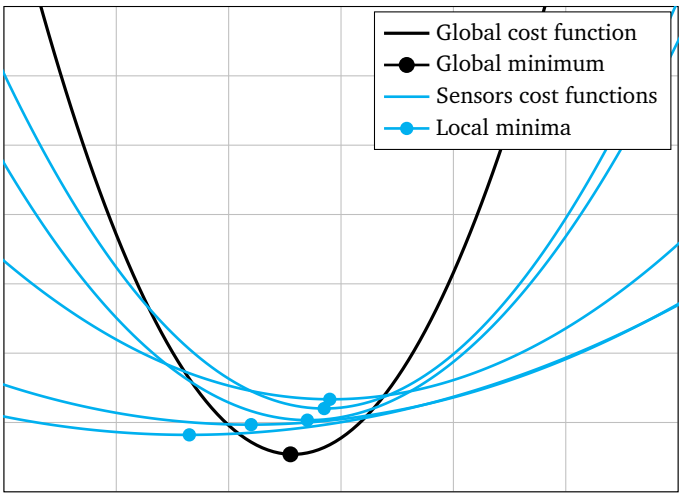


Figure 4.9: Local and global cost functions.

is perfectly compatible with any iterations-based stopping criterion.

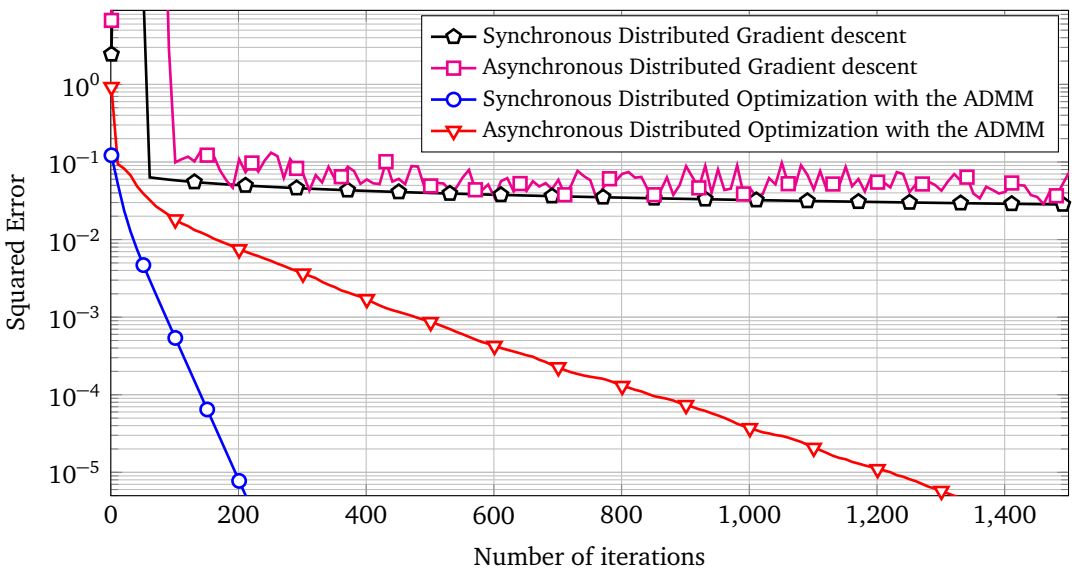


Figure 4.10: Squared Error versus the number iteration for various distributed optimization algorithms.

In Figs. 4.11 and 4.12, we plot the Squared Error versus the number of proximal operator computations and versus the number of communications for the two ADMM-based algorithms. The synchronous version requires 5 proximal operations and 10 communications per iteration while the asynchronous version uses 2 proximal operations and 3 communications. This enables us to see that when considering the computational time (of which the number of proximal operator computations is a fair approximation) or the network usage (represented by the number of local communications), then the difference between the synchronous and the asynchronous version is less important. Furthermore, if one adds the congestion of the

network on the parts where the subgraphs overlap and the differences in the computation times of the different proximal operators, the asynchronous version seems to be better suited to a decentralized network. In order to compare more precisely these two version, a general model for a network of agents including computation times, network congestion and so on would be useful. Designing such a model and analyzing the benefits of asynchronism is a very interesting perspective.

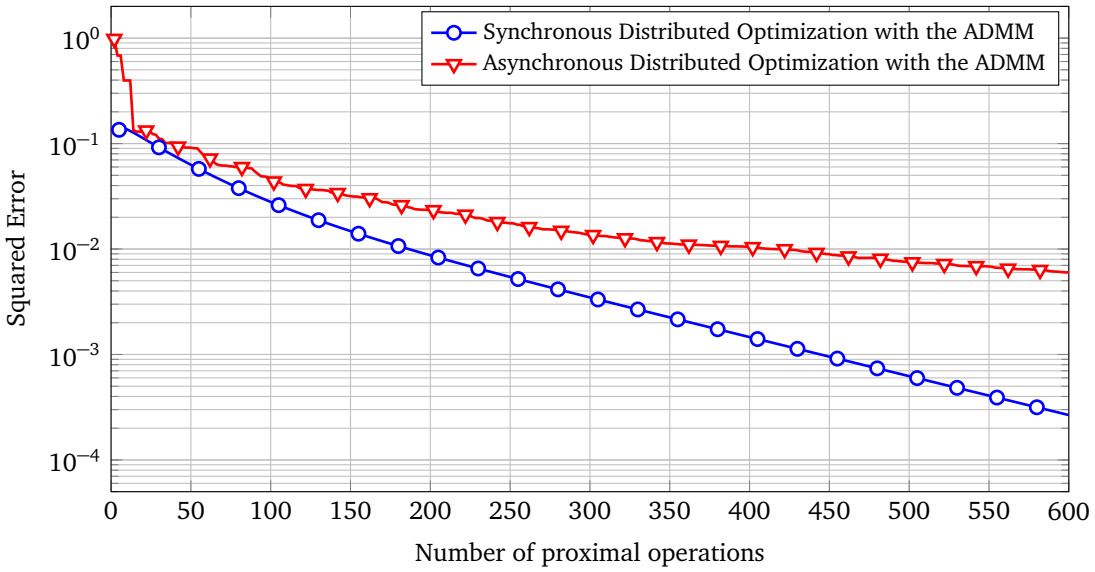


Figure 4.11: Squared error versus the number of proximal operator computations for ADMM-based algorithms.

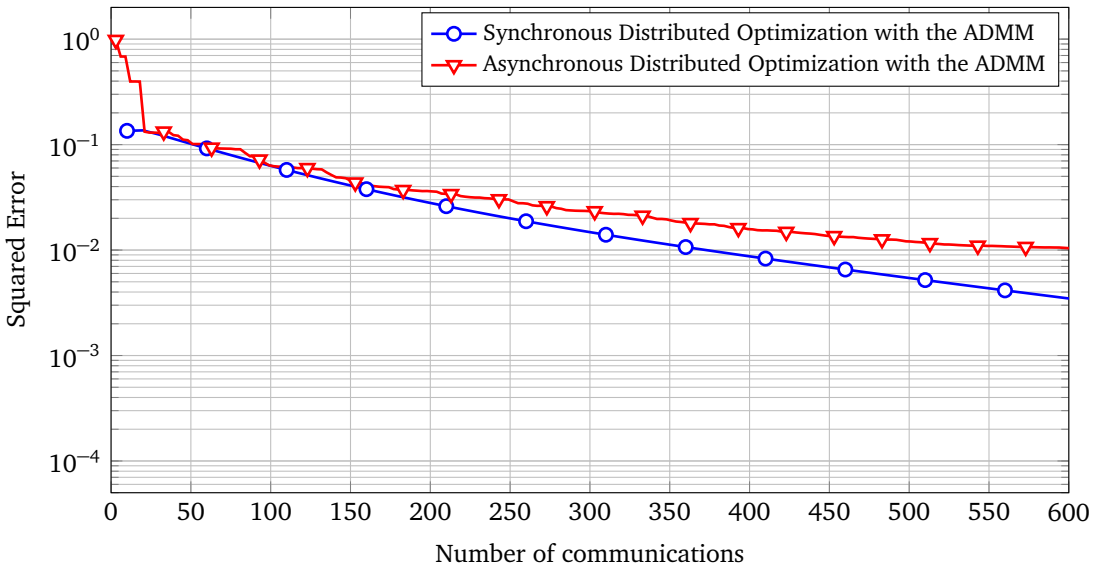


Figure 4.12: Squared error versus the number of communications for ADMM-based algorithms.

4.7 Conclusion

In this chapter, we introduced a formalism that enabled us to derive a new asynchronous optimization algorithm based of the well-known Alternating Direction Method of Multipliers and proved its convergence using a new randomized version of the proximal point algorithm. In a sensor network, this algorithm permits to obtain efficiently the wanted optimum by only using local data and asynchronous communications.

This topic is flourishing and has a major importance in the signal processing for Big Data and Social Networks; it is thus at the core of our research perspectives. For instance, finding tight speed bounds for ADMM-based algorithm is still an issue along with the hyper-parameter choice. Another promising axis of research is the study of the benefits of an asynchronous, local algorithm as the one we proposed in the case of an server network subject to congestion, limited computational times, which thus implies the definition of a global network metric.

Finally, we are often more interested to the optimal solution up to a small (freely selectable) error and the quicker an algorithm can ensure to be this close to the solution, the better it is. The costly step in our proposed algorithm is the computation of the proximal operator as it implies a small optimization subproblem. While it is known (from [81]) that one can compute the proximal operator up to a summable error over time without affecting the convergence, recent works [88, 89] seem to indicate that allowing a constant (possibly random) error at each step enable a faster convergence to a approximate solution.

This work has led to the following publications:

C4 **F. Iutzeler**, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers," in *IEEE Conference on Decision and Control (CDC)*, December 2013.

Conclusion and Perspectives

The work carried out in this thesis dealt with the analysis of distributed asynchronous algorithms that enabled the network to reach consensus over a value of interest. Depending on the objective value (maximum, average, or solution of an optimization problem), various algorithms have been proposed and analyzed.

In Chapter 2, we focused on the problem of finding and spreading the maximum between the initial values of the sensors; this problem can be cast into the rumor spreading framework. For this problem, we designed and analyzed three different algorithms: one based on a random walk, one based on pairwise communications, and one based on broadcast communications. We reminded the convergence speed results of literature for the first one and performed a new analysis of the convergence speed for the last two ones. The conclusions of this chapter are twofold: i) broadcasting significantly increases the speed of information spreading; and ii) we roughly pay a factor of about the size of the network in the mean convergence speed for not knowing which nodes have the information (maximum value or rumor).

In Chapter 3, we concentrated on the well-known problem of computing the average of the initial values of a sensor network. Although extensively studied, broadcast communications are allowed by only a few algorithms of the literature, especially, algorithms based on the so-called Sum-Weight framework. We proved very general convergence and convergence speed results for Sum-Weight based averaging algorithms; this enabled us to derive a new broadcast-based averaging algorithm which outperforms existing ones. Furthermore, our convergence results enabled us to revisit the literature and derived tighter speed bounds for existing algorithms. Finally, we used our new algorithm in the context of distributed spectrum sensing for cognitive radio.

In Chapter 4, we studied the problem of distributed asynchronous optimization using proximal algorithms. Thanks to the monotone operators framework, we designed a new distributed asynchronous optimization algorithm based on the Alternating Direction Method of Multipliers, and proved its convergence.

Perspectives

Chapters 2 and 3 have been very important for the comprehension of information spreading in radio networks and distributed estimation but even though there is some work left (finding

tighter speed bounds for maximum value spreading, analyzing the behavior of averaging algorithms in finite time with some error tolerance), it is not a priority for our future researches.

In contrast, Chapter 4 gave a very powerful analysis tool and our asynchronous proximal point algorithm has showed to perform remarkably well. Now, looking at the applications of algorithms such as our asynchronous distributed version of the ADMM, two very different setups arise depending if the datasets (and thus the cost functions) of the different nodes are similar or not: i) if they are similar (for example if each set contains samples from the same i.i.d. process), then the objective is to go to the better precision as fast as possible as a quick approximate result can be obtained at each node by analyzing only its own dataset; ii) if they are different (each set represent a different mixture of the same random variables) the goal of the network may be to obtain a snapshot of the global (mean) optimum as quickly as possible. These problems are very different and lead to different perspectives:

- In case of similar data, where we are interested in the asymptotic speed, a interesting perspective is to give mild conditions for the linear convergence of the ADMM; conditions for the linear convergence and speed bounds exist in the literature but for parallel (not distributed nor asynchronous) ADMM. Furthermore, the conditions are quite strong and the speed bounds are not tight. The choice of the hyper-parameter giving the best speed is also an open subject even if the convergence is linear as the speed bounds are not tight enough to deduce an optimal parameter; finding an efficient way to distributively select a good parameter is a very interesting perspective too. Also, the convergence speed could benefit from allowing the proximal operators to be computed up to some error (that would decrease over time) as it is in general quicker to compute the proximal operator up to some error. Finally, using the formalism depicted in Chapter 4 to analyze other proximal algorithms of the literature (PPXA, PPXA+) and compare their speed could lead to even more efficient and distributed algorithms.
- In case of heterogeneous data, we are interested in going to a neighborhood of the objective value as soon as possible. As the proximal operator can be very costly in term of computation time, it is interesting to look at the error to which a node can compute its proximal operator up to some error which still enable the network to converge to the wanted solution neighborhood. Indeed, computing a proximal operator is an optimization problem in itself and allowing an error in its computation reduces greatly the convergence time in general [88, 89].

These perspectives are mainly based on the number of iterations and the computational cost of the proximal operators to evaluate the speed of an algorithm. In particular, they do not take into account the effects of the network (congestion, failures, etc.). It would thus be very interesting to implement various synchronous and asynchronous distributed optimization algorithms over a real data network to perform inference through local computations. It would enable us to evaluate the influence of the asynchronism on the network congestion. In addition, it might guide us to define a global network metric including the computation times of the machines along with the network communications, routing and eventual congestion.

Appendices

Appendix A

Proofs related to Chapter 2

A.1 Proof of Eq. (2.3)

This proof follows the argument of [2].

Let $\mathcal{G} = (V, E)$ be an undirected graph of N nodes with Laplacian $\mathbf{L}$ and S a subset of V. We want to prove a relation between the second smallest eigenvalue of the Laplacian $\lambda_2^{\mathbf{L}}$ and the vertex expansion of S, $|\partial S|$, where $\partial S = \{\{i, j\} \in E : i \in S, j \notin S\}$ is the set of edges with one end in S and the other end in $V \setminus S$.

First, let us remark that when we compute $\mathbf{L}x$ for any $x \in \mathbb{R}^N$, we get that the i-th coefficient is

$$(\mathbf{L}x)_i = |\mathcal{N}_i| x_i - \sum_{j \in \mathcal{N}_i} x_j = \sum_{j \in \mathcal{N}_i} (x_i - x_j),$$

so we get that for any $x \in \mathbb{R}^N$

$$\begin{aligned} x^{\mathrm{T}}\mathbf{L}x &= \sum_{i \in V} x_i \sum_{j \in \mathcal{N}_i} (x_i - x_j) \\ &= \sum_{\{i,j\} \in E} x_i(x_i - x_j) + x_j(x_j - x_i) \\ &= \sum_{\{i,j\} \in E} (x_i - x_j)^2. \end{aligned}$$

Now, let us consider χ_S the size-N vector with ones at the index of the sensors of S and zeros elsewhere, we get that

$$\chi_S{}^{\mathrm{T}}\mathbf{L}\chi_S = \sum_{\{i,j\} \in E} (\chi_{Si} - \chi_{Sj})^2 = |\partial S|.$$

We also know that the second smallest eigenvalue of $\mathbf{L}$ is (see Section 1.1.1)

$$\lambda_2^{\mathbf{L}} = \min_{x \in \mathrm{Sp}(\mathbb{1})^{\perp}} \frac{x^{\mathrm{T}}\mathbf{L}x}{x^{\mathrm{T}}x}.$$

As $\chi_S \notin \mathrm{Sp}(\mathbb{1})^\perp$, we define $\overline{\chi_S} = \chi_S - |S|/N\mathbb{1}$, the orthogonal projection of χ_S onto $\mathrm{Sp}(\mathbb{1})^\perp$, we obviously have $\overline{\chi_S}^{\mathrm{T}}\mathbf{L}\overline{\chi_S} = {\chi_S}^{\mathrm{T}}\mathbf{L}\chi_S = |\partial S|$ and

$$\begin{aligned}\overline{\chi_S}^{\mathrm{T}}\chi_S &= \left(\chi_S - \frac{|S|}{N}\mathbb{1}\right)^{\mathrm{T}}\left(\chi_S - \frac{|S|}{N}\mathbb{1}\right)\\ &= \chi_S^{\mathrm{T}}\chi_S - \frac{|S|}{N}\mathbb{1}^{\mathrm{T}}\chi_S - \frac{|S|}{N}\chi_S^{\mathrm{T}}\mathbb{1} + \frac{|S|^2}{N^2}\mathbb{1}^{\mathrm{T}}\mathbb{1}\\ &= |S| - 2\frac{|S|}{N}|S| + \frac{|S|^2}{N} = |S|\left(1 - \frac{|S|}{N}\right)\end{aligned}$$

Hence,

$$\lambda_2^{\mathbf{L}} \leq \frac{\overline{\chi_S}^{\mathrm{T}}\mathbf{L}\overline{\chi_S}}{\overline{\chi_S}^{\mathrm{T}}\chi_S} = \frac{|\partial S|}{|S|\left(1 - \frac{|S|}{N}\right)}$$

and thus

$$\frac{|\partial S|}{|S|} \geq \lambda_2^{\mathbf{L}}\left(1 - \frac{|S|}{N}\right).$$

A.2 Proof of Theorem 2.4

As in the proof of Theorem 2.1, we define $X^k = |M^k|$ as the cardinal of $M^k = \{i \in V : x_i^k = x_{\max}\}$ (the set of nodes sharing the maximum at time k).

In the context of Random-Pairwise-Max, the only possibility for node to be informed is to exchange with an informed neighbor, so at each iteration one has $X^k \leq X^{k+1} \leq X^k + 1$. Here, our objective is to exhibit a tight evaluation of the probability that the sequence X^k is strictly increasing at time k. Let i and j be the selected (connected) sensors exchanging at iteration k of the algorithm, the size of M^k increases if i is informed but not j (or *vice-versa*),

$$\mathbb{P}[X^{k+1} = X^k + 1 \mid M^k] = \mathbb{P}[\{i,j\} \in \partial M^k | M^k]. \tag{A.1}$$

The choice of the exchanging sensors is done as follows: choose i uniformly in V then j uniformly in $\mathcal{N}_i$, all this is independent of the time so we drop the time superscript when unnecessary in the following. Therefore, for any (undirected) edge $\{i,j\}$, we have $\mathbb{P}[\{i,j\} \text{ are chosen}] \geq 2/(Nd_{\max})$ which implies that if i and j are the selected nodes at time k,

$$\mathbb{P}[\{i,j\} \in \partial M^k \mid M^k] \geq 2\frac{|\partial M^k|}{Nd_{\max}}. \tag{A.2}$$

Now, using Eq. (2.3) and combining Eqs. (A.1) and (A.2), we get

$$\mathbb{P}[X^{k+1} = X^k + 1 \mid M^k] \geq 2\frac{\lambda_2^{\mathbf{L}}}{Nd_{\max}}X^k(1 - \frac{X^k}{N}). \tag{A.3}$$

For the sake of simplicity, we assume that a single node has the maximum value at time 0 without loss of generality. Consider the stopping times: $\tau_i = \inf\{k \in \mathbb{N} : X^k = i\}$, so that

$\tau_1 = 0$ and $\tau = \sum_{i=1}^{N-1}(\tau_{i+1} - \tau_i)$ (if more than one node has the maximum value at time 0, one just has to start the sum at the number of initially informed nodes).

Let Y^k be the random variable equal to $X^{k+1} - X^k$ given X^k. Y^k is Bernoulli distributed and its parameter is lower bounded by $2\lambda_2^{\mathbf{L}}/(N^2 d_{\max})X^k(N - X^k)$ by Eq. (A.3).

$(\tau_{i+1} - \tau_i)$ is the number of iterations needed to have $X^{\tau_{i+1}} = X^{\tau_i} + 1$ when $X^{\tau_i} = i$, or equivalently, the number of trials on Y^{τ_i} to obtain a success when $X^{\tau_i} = i$. The random variable $(\tau_{i+1} - \tau_i)$ is thus bounded below by the geometrically distributed random variable Y^{τ_i}. As a consequence,

$$\mathbb{E}[\tau_{i+1} - \tau_i] \leq \frac{N^2 d_{\max}}{2\lambda_2^{\mathbf{L}}} \frac{1}{i(N-i)}.$$

Finally, using the fact that $2/N \sum_{i=1}^{N-1} 1/i = \sum_{i=1}^{N-1} 1/(i(N-i))$, we have

$$\mathbb{E}[\tau] \leq \frac{N d_{\max}}{\lambda_2^{\mathbf{L}}} \sum_{i=1}^{N-1} \frac{1}{i}.$$

A.3 Proof of Theorem 2.6

We use the same notations as in Appendix A.2. We have seen that the random variable $\tau_{i+1} - \tau_i$ is bounded below by a geometrically distributed of parameter $\pi_i = 2\lambda_2^{\mathbf{L}}/(N^2 d_{\max})i(N-i)$. As a consequence, $\tau_{i+1} - \tau_i$ is stochastically dominated by a geometric distribution with parameter π_i denoted by Z_i, which means that the cdf of $\tau_{i+1} - \tau_i$ is smaller than the cdf of Y_i at any point:

$$\mathbb{P}\left[\tau_{i+1} - \tau_i \geq (1+\delta)\frac{1}{\pi_i}\right] \leq \mathbb{P}\left[Z_i \geq (1+\delta)\frac{1}{\pi_i}\right].$$

Now, as Z_i is geometrically distributed, it can be seen as the first occurrence of a success in a process of i.i.d. Bernoulli variables $\{B_j\}_{j>0}$ of parameter π_i, thus we have, for any $\delta > 0$,

$$\begin{aligned}
\mathbb{P}\left[Z_i \geq (1+\delta)\frac{1}{\pi_i}\right] &\leq \mathbb{P}\left[\sum_{j=0}^{\lceil (1+\delta)\frac{1}{\pi_i} - 1 \rceil} B_j = 0\right] = (1-\pi_i)^{\lceil (1+\delta)\frac{1}{\pi_i} - 1 \rceil} \\
&\leq \exp\left(-\pi_i \lceil (1+\delta)\frac{1}{\pi_i} - 1 \rceil\right) \\
&\leq \exp\left(-(1+\delta) + \pi_i\right) \\
&\leq \exp(-\delta)
\end{aligned}$$

where the second inequality comes from the fact that $\forall x \in \mathbb{R},\ 1 + x \leq \exp(x)$ and the last inequality just uses that $\pi_i - 1 \leq 0$.

Let $\varepsilon > 0$, we now choose δ_ε as the smallest number such that

$$\exp(-\delta_\varepsilon) \leq \frac{\varepsilon}{N}$$

which leads to

$$\delta_\varepsilon = \log\left(\frac{N}{\varepsilon}\right).$$

So, we have $\mathbb{P}[\tau_{i+1} - \tau_i \geq (1+\delta_\varepsilon)/\pi_i] \leq \varepsilon/N$. Then, by using the Union bound, we have

$$\mathbb{P}\left[\tau \geq (1+\delta_\varepsilon)\mathbb{E}[\tau]\right] \leq \varepsilon$$

which concludes the proof.

A.4 Proof of Theorem 2.7

Let $i^{(0)}$ be the sensor with the maximum at time 0. We build a spanning tree subgraph of $\mathscr{G}$ rooted on $i^{(0)}$. Let us partition the set V in layers according to nodes distances from $i^{(0)}$:

$$L^n = \{i \in V : l^{\mathscr{G}}(i^{(0)}, i) = n\}, \quad n \in \mathbb{N}$$

where $l^{\mathscr{G}}(i^{(0)}, i)$ is the distance between $i^{(0)}$ and i (the minimal number of edges to connect them). One has $V = \cup_{n=0}^{\Delta_{\mathscr{G}}} L^n$ and $L^n \cap L^m = \emptyset$ for $n \neq m$. In the spanning tree subgraph, there are edges only between consecutive layers, never between sensors of a same layer nor between more distant layers. Furthermore, *all* the sensors of layer L^n have to broadcast in order to inform the whole layer L^{n+1}.

We define the following stopping times: $\tau_0 = 0$, and $\tau_n = \inf\{k \geq \tau_{n-1} : \forall i \in L^n, x_i^k = x_{\max}\}$. We denote by $\mathscr{F}_k$ the σ-algebra spanned by the nodes sharing the maximum values at time k. Using the same proof framework as in the standard coupon collector problem (see [34]), it is easy to show that $\mathbb{E}[\tau_{n+1} - \tau_n | \mathscr{F}_{\tau_n}] \leq N h_{|L^n|}$. More precisely, we want the expected time for L sensors/coupons to be chosen/collected knowing that any sensor/coupon is selected independently with probability $1/N$. The probability of selecting a *new* coupon knowing that $L - \ell$ ($\ell = 1, ..., L$) were already selected is ℓ/N, hence time for tis event to happen follow a geometric distribution of mean N/ℓ. Iterating this for $\ell = 1, ..., L$, we get that the expected time to select L coupons uniformly in a batch of size N is $N \sum_{\ell=1}^{L} 1/\ell$.

The term $\mathbb{E}[\tau_{n+1} - \tau_n | \mathscr{F}_{\tau_n}]$ corresponds to the average time needed to completely fill up layer $(n+1)$ given the nodes sharing the maximum value at time τ_n, *i.e.* given that layer n was already filled up. Therefore we have

$$\mathbb{E}[\tau] \leq \sum_{n=0}^{\Delta_{\mathscr{G}}-1} \mathbb{E}[\tau_{n+1} - \tau_n | \mathscr{F}_{\tau_n}] \leq \sum_{n=0}^{\Delta_{\mathscr{G}}-1} N h_{|L^n|}\,.$$

By using the inequality $h_n \leq \log(n) + 1$ and the fact $|L_0| = 1$, we obtain

$$\mathbb{E}[\tau] \leq N\left(\Delta_{\mathscr{G}} + \sum_{n=1}^{\Delta_{\mathscr{G}}-1} \log|L^n|\right).$$

Finally, using the concavity of the log, we get $\sum_{i=1}^{n-1} \log x_i \leq (n-1)\log(\frac{1}{n-1}\sum_{i=1}^{n-1} x_i)$, and remarking that $\sum_{n=1}^{\Delta_{\mathscr{G}}-1} |L^n| \leq N - 1$ (we dropped the sensor of the first layer in the previous

equation as $\log(1) = 0$), we get

$$\mathbb{E}[\tau] \leq N\Delta_{\mathscr{G}} + N(\Delta_{\mathscr{G}} - 1)\log\left(\frac{N-1}{\Delta_{\mathscr{G}} - 1}\right).$$

A.5 Proof of Theorem 2.8

Let $A_{i,n}^k$ be the event "the node i belonging to layer L^n is not chosen for k iterations". Obviously, $\mathbb{P}[A_{i,n}^k] = \left(1 - \frac{1}{N}\right)^k$. When the event $\tau_{n+1} - \tau_n \geq k$ occurs, the event $\cup_{i \in L^n} A_{i,n}^k$ also occurs. Therefore $\mathbb{P}[\tau_{n+1} - \tau_n \geq k] \leq \mathbb{P}[\cup_{i \in L^n} A_{i,n}^k]$. Using the Union bound and the fact that $0 \leq 1 - y \leq \exp(-y)$ for $y \in [0,1]$, one can bound the probability that, after k iterations, some of the nodes of L^n still have not talked as follows

$$\mathbb{P}[\tau_{n+1} - \tau_n \geq k] \leq \mathbb{P}[\cup_{i \in L^n} A_{i,n}^k] \leq \sum_{i \in L^n} \left(1 - \frac{1}{N}\right)^k \leq |L^n| \exp\left(-\frac{k}{N}\right).$$

For any $\varepsilon > 0$, by choosing $k_{\varepsilon,n} = N \log|L^n| + N \log(\Delta_{\mathscr{G}}/\varepsilon)$, we get

$$\mathbb{P}\left[\tau_{n+1} - \tau_n \geq N \log|L^n| + N \log\left(\frac{\Delta_{\mathscr{G}}}{\varepsilon}\right)\right] \leq \frac{\varepsilon}{\Delta_{\mathscr{G}}}.$$

By considering the union of the events for $n = 0, ..., \Delta_{\mathscr{G}} - 1$ and using once again the Union bound, we find

$$\mathbb{P}\left[\tau_{n+1} - \tau_n \geq N(\Delta_{\mathscr{G}} - 1)\log\left(\frac{N-1}{\Delta_{\mathscr{G}} - 1}\right) + N\Delta_{\mathscr{G}} \log\left(\frac{\Delta_{\mathscr{G}}}{\varepsilon}\right)\right] \leq \varepsilon$$

which boils down to the claimed result.

Appendix B

Proofs related to Chapter 3

B.1 Derivations for Eq. (3.38)

According to Eq. (3.37), we have easily that

$$\begin{aligned}\mathbf{K}^{\mathrm{T}}_{j_1^{k+1},\ldots,j_N^{k+1}}\mathbf{K}_{j_1^{k+1},\ldots,j_N^{k+1}} &= \frac{1}{4}\mathbf{I}+\frac{1}{4}\sum_{i=1}^{N}e_i e^{\mathrm{T}}_{j_i^{k+1}}+\frac{1}{4}\sum_{i=1}^{N}e_{j_i^{k+1}}e_i^{\mathrm{T}}+\frac{1}{4}\sum_{i=1}^{N}\sum_{i'=1}^{N}e_i e^{\mathrm{T}}_{j_i^{k+1}}e_{j_{i'}^{k+1}}e_{i'}^{\mathrm{T}}\\ &= \frac{1}{2}\mathbf{I}+\frac{1}{4}\sum_{i=1}^{N}e_i e^{\mathrm{T}}_{j_i^{k+1}}+\frac{1}{4}\sum_{i=1}^{N}e_{j_i^{k+1}}e_i^{\mathrm{T}}+\frac{1}{4}\sum_{i=1}^{N}\sum_{\substack{i'=1\\ i'\neq i}}^{N}e_i e^{\mathrm{T}}_{j_i^{k+1}}e_{j_{i'}^{k+1}}e_{i'}^{\mathrm{T}}\end{aligned}$$

where we remarked that the case $i=i'$ in the first equality leads to $1/4\mathbf{I}$ in the last right hand term. Now, using the fact that the $\{j_i^k\}_{i=1,\ldots,N;k>0}$ are i.i.d. uniformly distributed over $\{1,\ldots,N\}$ we have

$$\begin{aligned}\mathbb{E}\left[\mathbf{K}^{\mathrm{T}}\mathbf{K}\right] &= \frac{1}{4}\mathbf{I}+\frac{1}{4}\sum_{i=1}^{N}e_i\left(\frac{1}{N}\sum_{\ell=1}^{N}e_\ell^{\mathrm{T}}\right)+\frac{1}{4}\sum_{i=1}^{N}\left(\frac{1}{N}\sum_{\ell=1}^{N}e_\ell\right)e_i^{\mathrm{T}}\\ &\quad+\frac{1}{4}\sum_{i=1}^{N}\sum_{\substack{i'=1\\ i'\neq i}}^{N}e_i\left(\frac{1}{N^2}\sum_{\ell,\ell'=1}^{N}e_\ell^{\mathrm{T}}e_{\ell'}\right)e_{i'}^{\mathrm{T}}\end{aligned}$$

By remarking that $e_k^{\mathrm{T}}e_{k'}=0$ as soon as $k\neq k'$, we have $\sum_{k,k'=1}^{N}e_k^{\mathrm{T}}e_{k'}=N$. Furthermore,

$$\text{as}\quad \sum_{k=1}^{N}e_k=\mathbb{1}\quad\text{and}\quad\sum_{i=1}^{N}\sum_{\substack{i'=1\\ i'\neq i}}^{N}e_i e_{i'}^{\mathrm{T}}=\mathbb{1}\mathbb{1}^{\mathrm{T}}-\mathbf{I}$$

$$\text{we obtain}\quad \mathbb{E}\left[\mathbf{K}^{\mathrm{T}}\mathbf{K}\right] = \left(\frac{1}{2}-\frac{1}{4N}\right)\mathbf{I}+\frac{3}{4}\mathbf{J}$$

It is then straightforward to obtain Eq. (3.38).

B.2 Derivations for Eq. (3.40)

Once again, according to Eq. (3.37), we have directly that

$$\begin{aligned}\mathbf{K}_{j_1^{k+1},\dots,j_N^{k+1}} \otimes \mathbf{K}_{j_1^{k+1},\dots,j_N^{k+1}} &= \frac{1}{4}\mathbf{I}\otimes\mathbf{I} + \frac{1}{4}\left(\sum_{i=1}^N e_{j_i^{k+1}} e_i^{\mathrm{T}}\right)\otimes\mathbf{I} + \frac{1}{4}\mathbf{I}\otimes\left(\sum_{i=1}^N e_{j_i^{k+1}} e_i^{\mathrm{T}}\right) \\ &\quad + \underbrace{\frac{1}{4}\left(\sum_{i=1}^N e_{j_i^{k+1}} e_i^{\mathrm{T}}\right)\otimes\left(\sum_{i'=1}^N e_{j_{i'}^{k+1}} e_{i'}^{\mathrm{T}}\right)}_{\triangleq\, \varsigma} \end{aligned} \tag{B.1}$$

Using the same technique as in Appendix B.1 , we obtain that

$$\mathbb{E}\left[\sum_{i=1}^N e_{j_i^{k+1}} e_i^{\mathrm{T}}\right] = \mathbf{J} \tag{B.2}$$

Thus, it just remains to evaluate $\mathbb{E}[\varsigma]$. Let us first remark that

$$\varsigma = \sum_{i=1}^N \sum_{\substack{i'=1 \\ i'\neq i}}^N e_{j_i^{k+1}} e_i^{\mathrm{T}} \otimes e_{j_{i'}^{k+1}} e_{i'}^{\mathrm{T}} + \sum_{i=1}^N e_{j_i^{k+1}} e_i^{\mathrm{T}} \otimes e_{j_i^{k+1}} e_i^{\mathrm{T}}.$$

As a consequence, we have

$$\begin{aligned}\mathbb{E}[\varsigma] &= \frac{1}{N^2}\sum_{i=1}^N \sum_{\substack{i'=1 \\ i'\neq i}}^N \sum_{\ell=1}^N \sum_{\ell'=1}^N e_\ell e_i^{\mathrm{T}} \otimes e_{\ell'} e_{i'}^{\mathrm{T}} + \frac{1}{N}\sum_{i=1}^N \sum_{\ell=1}^N e_\ell e_i^{\mathrm{T}} \otimes e_\ell e_i^{\mathrm{T}} \\ &= \frac{1}{N^2}\sum_{i=1}^N \sum_{i'=1}^N \sum_{\ell=1}^N \sum_{\ell'=1}^N e_\ell e_i^{\mathrm{T}} \otimes e_{\ell'} e_{i'}^{\mathrm{T}} - \frac{1}{N^2}\sum_{i=1}^N \sum_{\ell=1}^N \sum_{\ell'=1}^N e_\ell e_i^{\mathrm{T}} \otimes e_{\ell'} e_i^{\mathrm{T}} + \frac{1}{N}\sum_{i=1}^N \sum_{\ell=1}^N e_\ell e_i^{\mathrm{T}} \otimes e_\ell e_i^{\mathrm{T}}\end{aligned}$$

Using the well-known result on Kronecker product ($(\mathbf{AB})\otimes(\mathbf{CD}) = (\mathbf{A}\otimes\mathbf{C})(\mathbf{B}\otimes\mathbf{D})$ for four matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ with appropriate sizes), we have

$$\mathbb{E}[\varsigma] = \mathbf{J}\otimes\mathbf{J} - \frac{1}{N^2}(\mathbb{1}\otimes\mathbb{1})u^{\mathrm{T}} + \frac{1}{N}uu^{\mathrm{T}} \tag{B.3}$$

with $u = \sum_{i=1}^N e_i \otimes e_i$. Putting Eqs. (B.2) and (B.3) into Eq. (B.1), we get

$$\mathbb{E}\left[\mathbf{K}\otimes\mathbf{K}\right] = \frac{1}{4}\mathbf{I}\otimes\mathbf{I} + \frac{1}{4}\mathbf{J}\otimes\mathbf{I} + \frac{1}{4}\mathbf{I}\otimes\mathbf{J} + \frac{1}{4}\mathbf{J}\otimes\mathbf{J} - \frac{1}{4N^2}(\mathbb{1}\otimes\mathbb{1})u^{\mathrm{T}} + \frac{1}{4N}uu^{\mathrm{T}}.$$

Before going further, let us remark that

$$\begin{aligned}u^{\mathrm{T}}(\mathbf{J}^\perp\otimes\mathbf{J}^\perp) &= \sum_{i=1}^N (e_i^{\mathrm{T}} - \frac{1}{N}e_i^{\mathrm{T}}\mathbb{1}\mathbb{1}^{\mathrm{T}})\otimes(e_i^{\mathrm{T}} - \frac{1}{N}e_i^{\mathrm{T}}\mathbb{1}\mathbb{1}^{\mathrm{T}}) = \sum_{i=1}^N (e_i^{\mathrm{T}} - \frac{1}{N}\mathbb{1}^{\mathrm{T}})\otimes(e_i^{\mathrm{T}} - \frac{1}{N}\mathbb{1}^{\mathrm{T}}) \\ &= \sum_{i=1}^N e_i^{\mathrm{T}}\otimes e_i^{\mathrm{T}} - \frac{1}{N}\sum_{i=1}^N (e_i^{\mathrm{T}}\otimes\mathbb{1}^{\mathrm{T}}) - \frac{1}{N}\sum_{i=1}^N (\mathbb{1}^{\mathrm{T}}\otimes e_i^{\mathrm{T}}) + \frac{1}{N^2}\sum_{i=1}^N \mathbb{1}^{\mathrm{T}}\otimes\mathbb{1}^{\mathrm{T}} \\ &= u^{\mathrm{T}} - \frac{1}{N}(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}.\end{aligned} \tag{B.4}$$

As a consequence, we have

$$\begin{aligned}\mathbf{R} &= \mathbb{E}[\mathbf{K}\otimes\mathbf{K}](\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}) \\ &= \frac{1}{4}\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp} - \frac{1}{4N^2}(\mathbb{1}\otimes\mathbb{1})u^{\mathrm{T}}(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}) + \frac{1}{4N}uu^{\mathrm{T}}(\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}) \\ &= \frac{1}{4}\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp} - \frac{1}{4N^2}(\mathbb{1}\otimes\mathbb{1})(u^{\mathrm{T}} - \frac{1}{N}(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}) + \frac{1}{4N}u(u^{\mathrm{T}} - \frac{1}{N}(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}}) \\ &= \frac{1}{4}\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp} + \frac{1}{4N}uu^{\mathrm{T}} - \frac{1}{4N^2}(\mathbb{1}\otimes\mathbb{1})u^{\mathrm{T}} - \frac{1}{4N^2}u(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}} + \frac{1}{4N}\mathbf{J}\otimes\mathbf{J}\end{aligned}$$

Using Eq. (B.4), we define $v = 1/\sqrt{N-1}(u - 1/N\mathbb{1}\otimes\mathbb{1})$ so that

$$vv^{\mathrm{T}} = \frac{1}{N-1}\left(uu^{\mathrm{T}} - \frac{1}{N}(\mathbb{1}\otimes\mathbb{1})u^{\mathrm{T}} - \frac{1}{N}u(\mathbb{1}\otimes\mathbb{1})^{\mathrm{T}} + \mathbf{J}\otimes\mathbf{J}\right)$$

which straightforwardly leads to Eq. (3.40).

In addition, note that using Eq. (B.4), we have $\mathbf{J}^{\perp}\otimes\mathbf{J}^{\perp}v = v$.

B.3 Computations related to Section 3.7.3-a

In order to fit a Gamma distribution onto it, one has to compute the mean and variance of

$$\mathrm{T}(y) \triangleq \frac{1}{N}\sum_{i=1}^{N}\frac{\|y_i\|_2^2}{\gamma_i^2+\sigma_i^2}\mathrm{SNR}_i \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta$$

with $x_i \sim \mathcal{N}(0, \gamma_i^2\mathbf{I})$ and $n_i \sim \mathcal{N}(0, \sigma_i^2\mathbf{I})$.

$$\begin{aligned}\mathbb{E}[\mathrm{T}(y)] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}.\mathrm{SNR}_i\right] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}\right]\mathrm{SNR}_i \\ &= \frac{N_s}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\end{aligned}$$

$$
\begin{aligned}
\mathrm{Var}[\mathrm{T}(y)] &= \mathbb{E}\left[\left(\sum_{i=1}^{N}\frac{1}{N}\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}.\mathrm{SNR}_i\right)^2\right]-\left(\frac{N_s}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{1}{N^2}\mathbb{E}\left[\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}\frac{\|x_j+n_j\|_2^2}{\gamma_j^2+\sigma_j^2}.\mathrm{SNR}_i\mathrm{SNR}_j\right]-\left(\frac{N_s}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\mathbb{E}\left[\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}\right]\mathbb{E}\left[\frac{\|x_j+n_j\|_2^2}{\gamma_j^2+\sigma_j^2}\right].\mathrm{SNR}_i\mathrm{SNR}_j\\
&\quad+\frac{1}{N^2}\sum_{i=1}^{N}\mathbb{E}\left[\left(\frac{\|x_i+n_i\|_2^2}{\gamma_i^2+\sigma_i^2}\right)^2\right]\mathrm{SNR}_i^2-\left(\frac{N_s}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{N_s^2}{N^2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\mathrm{SNR}_i\mathrm{SNR}_j+\frac{2N_s+N_s^2}{N^2}\sum_{i=1}^{N}\mathrm{SNR}_i^2-\frac{N_s^2}{N^2}\left(\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{N_s^2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{SNR}_i\mathrm{SNR}_j+2\frac{N_s}{N^2}\sum_{i=1}^{N}\mathrm{SNR}_i^2-\frac{N_s^2}{N^2}\left(\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{N_s^2}{N^2}\left(\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2+2\frac{N_s}{N^2}\sum_{i=1}^{N}\mathrm{SNR}_i^2-\frac{N_s^2}{N^2}\left(\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2\\
&= \frac{2N_s}{N^2}\sum_{i=1}^{N}\mathrm{SNR}_i^2
\end{aligned}
$$

The gamma distribution $\Gamma(\kappa,\theta)$ has mean $\kappa\theta$ and variance $\kappa\theta^2$. Hence, by identifying the first two moments of $\mathrm{T}(y)$, we get that $\mathrm{T}(y)\approx\Gamma(\kappa_\mathrm{T},\theta_\mathrm{T})$ with

$$
\begin{aligned}
\kappa_\mathrm{T} &= \frac{NN_s}{2}\cdot\frac{\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2}\\
\text{and}\quad \theta_\mathrm{T} &= \frac{2}{N}\cdot\frac{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i}.
\end{aligned}
$$

RÉSUMÉ EN FRANÇAIS

INTRODUCTION

Les travaux présentés dans cette thèse ont été conduits dans le groupe Communications Numériques, département COMmunications et ÉLECtronique (COMELEC) de Telecom ParisTech entre Octobre 2010 et Octobre 2013. Ces travaux ont été financés par la Direction Générale de l'Armement (DGA) et l'Institut Carnot Telecom/EURECOM.

Description du Problème

Depuis une dizaine d'années, l'estimation/détection/optimisation distribuée a été le sujet d'un grand nombre de travaux provenant de communautés très différentes. En effet, calculer une valeur globale dans un réseau à partir d'informations locales des agents et des liens qui existent entre eux a de nombreuses applications allant des réseaux de capteurs au *cloud computing* en passant par l'apprentissage automatique.

Par exemple, dans de nombreux réseaux *big data*, l'information à traiter est généralement localisée à des endroits physiquement distants et d'une quantité telle qu'il est impossible pour une seule machine de la traiter entièrement. Il est donc intéressant de considérer un réseau d'agents possédant des quantités d'information assez petites pour pouvoir être traitées entièrement ; afin d'obtenir le résultat du traitement sur l'ensemble des données, ces agents doivent communiquer intelligemment. Aussi, pour éviter une congestion du réseau et la présence d'un ordonnanceur global, il est préférable que les agents communiquent localement de manière asynchrone.

Ainsi, dans cette thèse, **nous considérons la tâche d'optimiser/calculer une valeur globale à partir de communications locales et asynchrones**, nous évitons ainsi la présence d'un centre de fusion. Évidemment les méthodes employées pour résoudre de tels problèmes dépendent grandement de la nature de la valeur cherchée, qui peut aller de la valeur d'un agent particulier du réseau (il suffit alors de la trouver puis de la propager) à la solution d'un problème d'optimisation complexe. Donc, afin de présenter un panorama quelque peu représentatif, nous considérons trois problèmes très différents :

- **Problème 1 :** Trouver et propager la plus grande des valeurs initiales dans un réseau de capteurs ; ce problème nous permettra de mieux comprendre la diffusion de l'information

dans les réseaux.

- **Problème 2 :** Calculer la moyenne des valeurs initiales des capteurs ; ce problème nous permettra de mieux comprendre comment combiner les informations du réseau de manière égalitaire.
- **Problème 3 :** Trouver l'optimum d'une fonction globale, définie comme la somme des fonctions locales des agents, alors que chaque agent n'a accès qu'à sa fonction locale.

Organisation de la thèse et Contributions

Avant de s'intéresser aux problèmes décrits plus haut, nous décrivons notre modèle dans la première partie. Plus précisément, nous introduisons notre modèle pour les réseaux de capteurs sans-fil où les agents et les canaux qui les relient sont modélisés par un graphe ; nous discutons également des conséquences et de la mise en pratique de l'hypothèse d'asynchronisme qui sera présente tout au long de la thèse. De plus, nous énonçons quelques propriétés essentielles des matrices positives et de leurs liens avec les graphes.

Dans la deuxième partie, nous nous intéressons au premier problème, c'est-à-dire trouver et propager le maximum des valeurs initiales d'un réseau de capteurs sans-fil. Soit un réseau de capteurs où chacun possède une valeur (scalaire) initiale, nous voulons que les agents du réseau communiquent à l'aide des liens sans-fil qui les relient (ce qui veut dire qu'ils peuvent *broadcaster* leur information) de manière à ce que la valeur initiale maximale soit connue par tous. Ce problème peut sembler artificiel mais il est utile afin de comprendre la propagation d'une information à travers un réseau. Notre problème est proche de la propagation de rumeurs (*rumor spreading*) dans un réseau sans-fil asynchrone, cependant, une différence importante existe entre les deux problèmes car dans le cadre du *rumor spreading*, les agents savent s'ils possèdent ou non la rumeur, ce qui n'est pas le cas ici. Nous analysons ce problème à travers deux différents algorithmes : un algorithme aléatoire basé sur des communications pair à pair et autre basé sur des communications *broadcast*. Contrairement à la marche aléatoire, les deux derniers algorithmes sont nouveaux et donc non traités dans la littérature. Notre contribution à ce problème est donc d'analyser en détail ces deux algorithmes en calculant des bornes sur l'espérance de leur temps de convergence et sur la dispersion de celui-ci. Ces nouveaux résultats nous permettent d'illustrer le gain en performances du à l'utilisation de communications de type *broadcast*.

Dans la troisième partie, nous étudions les algorithmes de moyennage. Soit un réseau de capteurs sans-fil où chacun possède une valeur (scalaire) initiale, nous voulons désormais trouver la moyenne des valeurs initiales de manière distribuée. Ainsi, les capteurs doivent échanger localement et de manière asynchrone et faire des combinaisons linéaires bien choisies entre leur valeur et les valeurs reçues. Les algorithmes de moyennage dans les réseaux sans-fil reçoivent beaucoup d'attention depuis une dizaine d'années. Un des plus célèbres algorithmes de la littérature est le *Random Gossip*, il consiste simplement à choisir aléatoirement une paire d'agents connectés à chaque itération puis de leur faire échanger et moyenner leurs valeurs. Des algorithmes de moyennage basés sur des communications de type *broadcast* ont égale-

ment été développés mais ils souffrent soit de problèmes de convergence soit de problèmes de collisions liées au besoin d'une voie de retour pour chacun des voisins contactés. A l'aide des propriétés rappelées dans la première partie, nous passons en revue les résultats de convergence (et de non-convergence) de la littérature. Ensuite, comme la deuxième partie a vanté les mérites de communications de type *broadcast*, nous considérons les algorithmes de moyennage basés sur des communications de type *broadcast* sans voie de retour[1]. Récemment, Bénézit *et al.* ont prouvé qu'en maintenant deux variables par capteur, une initialisée avec la valeur initiale du capteur et l'autre à 1, et en les mettant simultanément à jour, il était possible de concevoir des algorithmes de moyennage sans voie de retour tels que le ratio de ces deux variables tend vers la moyenne voulue. En fait, ce cadre particulier basé sur deux variables mises à jour simultanément s'appelle *Sum-Weight* et a été introduit par Kempe en 2003. Les principales contributions de cette partie sont donc : i) la création d'un nouvel algorithme de moyennage basé sur des communications *broadcast* sans voie de retour appelé *Broadcast Weighted Gossip* (*BWGossip*) et ii) la preuve que les algorithmes basés sur le *Sum-Weight* convergent à une vitesse exponentielle sous des hypothèses faibles (une borne sur cette vitesse est également calculée). Il est à noter que ces résultats s'appliquent également aux algorithmes de moyennage existants (avec voie de retour et une variable par capteur). Aussi, nous montrons par simulation que notre borne sur la vitesse est très fine autant pour notre algorithme que pour ceux de la littérature. Finalement, nous remarquons qu'une version modifiée du *Sum-Weight* nous permet de calculer la somme des valeurs initiales du réseau sans en connaître le nombre d'agents. Comme application de ces résultats, nous considérons le problème de la détection de présence dans un système de radio cognitive. En effet, les utilisateurs secondaires ont intérêt à collaborer afin de détecter si un utilisateur primaire utilise la ressource ou non. Nous montrons que ce problème repose sur le calcul d'un moyenne et d'une somme des résultats des tests des utilisateurs secondaires ; donc, à l'aide des résultats précédents, nous avons pu créer un nouvel algorithme de détection totalement distribué.

Dans la quatrième et dernière partie, nous cherchons à résoudre de manière asynchrone un problème d'optimisation distribuée, c'est-à-dire un problème dans lequel un réseau veut connaître le minimum de la somme de fonctions de coût de ses agents. Cette situation est courante dans les réseaux *big data* où la quantité d'informations stockées sur des machines physiquement distantes est bien trop grande pour être acheminée vers un centre de fusion. Ainsi, pour traiter cette énorme masse de données, les agents du réseau doivent traiter intelligemment leurs données locales et échanger quelques informations (quelques scalaires tout au plus) de manière asynchrone avec d'autres agents relativement proches dans le réseau (afin de ne pas noyer le réseau à cause d'informations à router). Soit f la fonction convexe, dépendant de l'ensemble des données, dont nous voulons connaître un minimum ; nous supposons que f peut s'écrire $\sum_i f_i$ où f_i est une fonction convexe dépendant uniquement des données de l'agent i. Notre problème est donc de trouver un minimum de f dans un réseau où chaque

[1] Nous permettons néanmoins la présence d'un canal de contrôle permettant à l'émetteur de savoir si le message a été reçu par les destinataires.

agent i a uniquement accès à f_i. Nous nous intéresserons d'abord aux algorithmes du premier ordre où chaque agent du réseau effectue une descente de gradient sur sa fonction propre puis le réseau effectue une étape de moyennage (étudiée dans la partie précédente). Ces algorithmes sont plutôt lents et n'utilisent qu'une partie de leur fonction de coût (le gradient en un point). Avec l'augmentation des capacités de calcul, il est donc pertinent de développer des algorithmes qui utilisent plus complètement les fonctions des agents au prix d'un coût en calcul plus élevé. Récemment, il a été montré qu'un algorithme appelé *Alternating Direction Method of Multipliers* (ADMM) était particulièrement bien adapté à l'optimisation distribuée dans un contexte synchrone. Nos principales contributions ont donc été de développer un algorithme d'optimisation distribuée asynchrone basé sur l'ADMM et de prouver sa convergence à l'aide du formalisme des opérateurs monotones.

Publications

Articles de Revues

J2 **F. Iutzeler**, P. Ciblat, and W. Hachem, "Analysis of Sum-Weight-like algorithms for averaging in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2802–2814, June 2013.

J1 **F. Iutzeler**, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 6103–6107, November 2012.

Conférences Internationales

C4 **F. Iutzeler**, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers," in *IEEE Conference on Decision and Control (CDC)*, December 2013.

C3 **F. Iutzeler** and P. Ciblat, "Fully-distributed spectrum sensing: application to cognitive radio," in *European Signal Processing Conference (EUSIPCO)*, September 2013.

C2 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "New broadcast based distributed averaging algorithm over wireless sensor networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012.

C1 **F. Iutzeler**, J. Jakubowicz, W. Hachem, and P. Ciblat, "Distributed estimation of the maximum value over a Wireless Sensor Network," in *Asilomar Conference on Signals, Systems and Computers*, November 2011.

Conférences Nationales

N2 **F. Iutzeler** and P. Ciblat, "Sondage de spectre coopératif totalement distribué: application à la radio cognitive," in *Colloque GRETSI* , September 2013.

N1 **F. Iutzeler**, P. Ciblat, W. Hachem, and J. Jakubowicz, "Estimation distribuée du maximum dans un réseau de capteurs," in *Colloque GRETSI*, September 2011.

I MODÈLE

Nous modélisons notre réseau de N agents par un graphe $\mathcal{G} = (V, E)$ avec V l'ensemble des nœuds/agents/capteurs et E l'ensemble des arêtes/liens parfaits entre les agents ; $(i, j) \in E$ veut ainsi dire qu'il y a un lien direct allant de l'agent i à l'agent j. Les voisins de i sont notés $\mathcal{N}_i = \{j \in V : (i, j) \in E\}$ et représentent les agents à qui i peut envoyer de l'information. Si les liens sont bidirectionnels, c'est-à-dire si $(i, j) \in E \Rightarrow (j, i) \in E$, nous dirons que le graphe est *non-dirigé* (il sera dit dirigé sinon). Dans notre contexte de diffusion d'information dans les réseaux, il est important de considérer la connexité d'un graphe. Un graphe est dit *faiblement connexe* si pour tout $i, j \in V$ il y a un chemin (une succession d'arêtes) de i à j **ou** de j à i ; il sera *fortement connexe* si les deux chemins existent pour tout couple de nœuds. Dans les graphes non-dirigés, les deux notions sont équivalentes et nous dirons juste *connexe*. Nous travaillerons principalement sur des graphes (fortement) connexes. Un agent i peut être connecté avec lui même, on dit alors qu'il est *auto-connecté*. Enfin, il est possible d'assigner un poids $\omega_{(i,j)} > 0$ à toute arête du graphe, le rendant ainsi *pondéré* (par défaut, ces poids sont pris égaux à 1). Ces deux propriétés ne changent évidemment pas la connexité du graphe.

Définissons maintenant quelques matrices importantes en théorie des graphes. D'une part, la *matrice d'Adjacence* $\mathbf{A}$ est la matrice carrée de taille N telle que $\mathbf{A}_{i,j} = \omega_{(i,j)}$ si $(i, j) \in E$ et 0 ailleurs. D'autre part, dans le cas d'un graphe non-dirigé[2], la *matrice Laplacienne* $\mathbf{L}$ est définie par $\mathbf{L}_{i,j} = -\omega_{(i,j)}$ si $(i, j) \in E$, $\mathbf{L}_{i,i} = \sum_{j \in \mathcal{N}_i} \omega_{(i,j)}$, et 0 ailleurs. Les valeurs propres de cette dernière jouent un rôle important dans la théorie algébrique des graphes. En effet, elles sont toutes positives, 0 est valeur propre (associée au vecteur $\mathbb{1} = [1, ..., 1]^{\mathrm{T}}$) et la seconde plus petite valeur propre $\lambda_2^{\mathbf{L}}$ est strictement positive si et seulement si le graphe est connecté [1].

Dans les réseaux de capteurs, les agents possèdent généralement une ou plusieurs *valeurs initiales* dont le réseau voudra tirer parti via la valeur maximale, moyenne, etc. Formellement, un capteur i possède une valeur initiale x_i^0 de la variable x, nous notons $x^0 = [x_1^0, ..., x_N^0]^{\mathrm{T}}$.

Nous travaillerons principalement avec des réseaux *asynchrones* dans lesquels une action correspond au réveil d'un ou d'un petit groupe d'agents qui vont alors échanger et calculer une nouvelle version de leurs variables personnelles. Nous prendrons comme échelle de temps le nombre d'actions si bien qu'au temps k, k actions se sont produites et la variable du capteur i s'écrira x_i^k, nous noterons aussi $x^k = [x_1^k, ..., x_N^k]^{\mathrm{T}}$.

[2] des définitions similaires existent dans le cas dirigé mais ne seront pas utiles dans la suite [5].

II PROPAGATION DISTRIBUÉE DU MAXIMUM

Nous étudions ici différentes techniques pour propager la plus grande des valeurs initiales d'un réseau sans-fil en utilisant des communications locales asynchrones.

II.1 Introduction

Les réseaux de capteurs sans-fil sont généralement déployés dans des environnements hostiles où aucun agent ne peut jouer le rôle de centre de fusion à cause de la faible connexion du réseau et du coût prohibitif des communications. Dans de tels réseaux, une action spéciale (mécanique, information à transmettre par un lien très coûteux, etc.) doit parfois être effectuée par un des agents, il est alors naturel de choisir l'agent avec la plus grande puissance ou la plus grande énergie restante pour effectuer cette action. Pour ceci, le réseau doit trouver et faire connaître l'identité de ce capteur de manière distribuée en utilisant de manière asynchrone les liens sans-fil entre eux. Cette action requiert donc le calcul distribué du maximum des valeurs initiales du réseau.

Ce problème est proche du *rumor spreading* (propagation de rumeur, d'épidémie) introduit par Demers *et al.* dans les années 1980 [10] dans le cas des bases de données répliquées (pages jaunes, DNS, etc.). Trois concepts importants y sont introduits : i) le *colportage de rumeur* qui est l'idée que certaines informations du réseau sont plus importantes que d'autres et que celles-ci méritent d'être transmises ; ii) les transmissions *push* où l'information va de l'appelant à l'appelé et iii) les transmissions *pull* où l'information va de l'appelé à l'appelant. A cause de la nature filaire des travaux originaux dans le domaine, la plupart des papiers de la littérature traitent de communications *push et/ou pull* entre deux agents, de manière synchrone [12, 13, 14, 15, 16, 17, 18, 19] cependant quelques papiers considèrent également des réseaux sans-fil et donc des communications *broadcast* [20, 21, 22, 23, 24].

Les différences entre le *rumor spreading* et notre propagation distribuée du maximum sont de deux natures. D'abord, dans le cadre du *rumor spreading*, les nœuds savent s'ils disposent de la rumeur à transmettre (*i.e.*du max) ou non, ce qui n'est pas le cas dans notre modèle. Deuxièmement, dans le cadre du *rumor spreading*, les communications sont synchrones et plusieurs agents transmettent simultanément, ce qui engendre des collisions; dans notre modèle, les communications sont asynchrones et un seul agent initie une transmission à chaque itération, il n'y a donc pas de collisions.

Bien que proche, notre problème est donc nouveau ([25] étudie le calcul distribué d'une grande variété de fonctions dont le maximum mais les mises à jours sont incrémentales et donc peu adaptées au calcul du maximum), bien distinct du *rumor spreading* et mérite donc une analyse particulière.

II.2 Algorithmes

Nous présentons ici deux algorithmes naturels résolvant notre problème. Le premier est

basé sur des communications pair à pair entre deux agents connectés tandis que le deuxième tire parti de la nature sans-fil des liens du réseau en utilisant une communication *broadcast*.

Algorithme: *Random-Pairwise-Max*. Une manière simple d'accomplir notre but est que l'agent qui s'active contacte un voisin au hasard, puis qu'ils échangent leurs valeurs et gardent la plus grande des deux (de manière similaire au *push et pull* du *Rumor spreading*).

Random-Pairwise-Max

Soit i l'agent actif au temps k.

- i choisit un voisin j uniformément et ils échangent leurs valeurs.
- i et j se mettent à jour : $x_i^{k+1} = x_j^{k+1} = \max\left(x_i^k, x_j^k\right)$.

Cet algorithme est adapté aux réseaux filaires mais n'est clairement pas optimal dans le cas des réseaux sans-fil car il n'utilise pas la nature *broadcast* des canaux.

Algorithme: *Random-Broadcast-Max*. Dans notre contexte de réseaux sans-fil, il semble naturel que le capteur actif *broadcast* sa valeur puis que les capteurs l'ayant reçu la conservent si elle est plus grande que la leur.

Random-Broadcast-Max

Soit i l'agent actif au temps k.

- i *broadcast* x_i^k à tous ses voisins.
- Les voisins se mettent à jour : $x_j^{k+1} = \max\left(x_j^k, x_i^k\right)$ pour tout $j \in \mathcal{N}_i$.

On peut remarquer que si le capteur actif n'a pas reçu d'information depuis la dernière fois qu'il a émis, l'itération est inutile et peut donc être abandonnée ce qui réduit le nombre de communications mais pas le temps de convergence, nous ne considérerons donc pas ce changement dans la suite.

Comparaison. La Figure II.1 représente le pourcentage moyen de capteurs ayant la valeur maximale en fonction du temps sur un graphe de 50 nœuds, nous remarquons que l'utilisation de communications *broadcast* accélère la vitesse de convergence par rapport aux communications pair à pair.

II.3 Performances

Soit $\tau \triangleq \inf\{t \in \mathbb{N} : \forall k \geq t \quad x^k = \max_i x_i^0\}$, le *temps de convergence* d'un algorithme. Nous avons analysé les performances de deux algorithmes décrits plus hauts à travers la moyenne de ce temps de convergence et une borne supérieure de ce temps avec probabilité $1-\varepsilon$. Nous nous sommes attachés à faire transparaître autant que possible les caractéristiques du graphe sous-jacent dans ces bornes. Les preuves des deux résultats suivants se trouvent dans l'Annexe A.

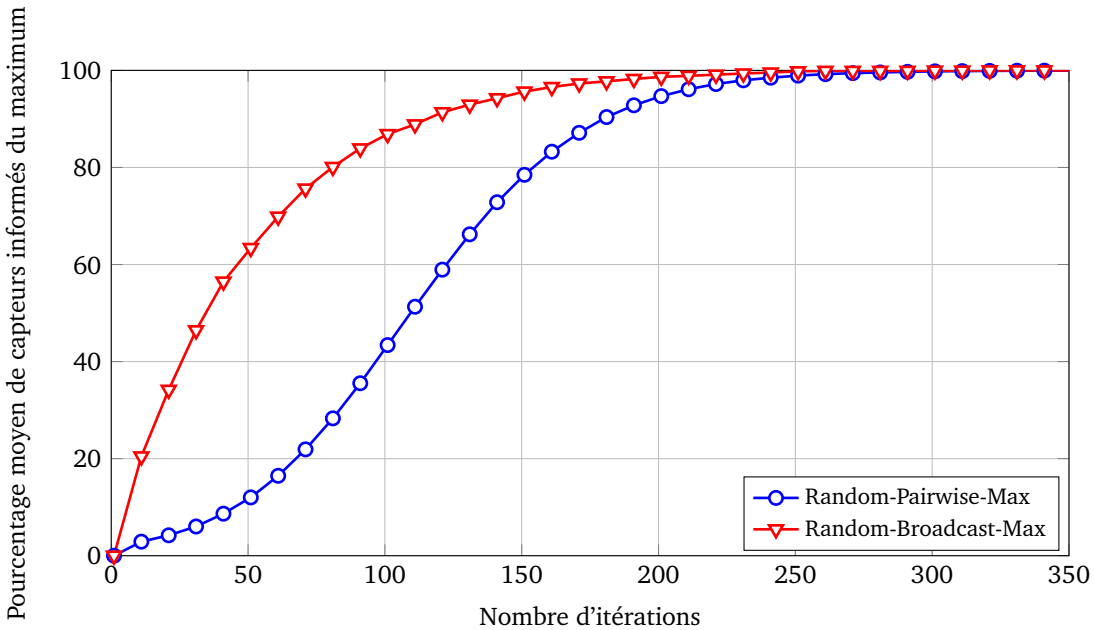


Figure II.1 : Pourcentage de capteurs ayant la valeur maximale en fonction du temps.

Théorème. *Pour le* RANDOM-PAIRWISE-MAX*, on a*

$$\begin{aligned} \mathbb{E}[\tau] &\leq N d_{\max} \frac{h_{N-1}}{\lambda_2^{\mathbf{L}}} \triangleq \alpha \\ et\ \tau &\leq \alpha + \alpha \log\left(\frac{N}{\varepsilon}\right) \text{ avec probabilité } 1-\varepsilon \end{aligned}$$

où $d_{\max}$ est le degré maximal, h_N est le N-ième nombre harmonique et $\lambda_2^{\mathbf{L}}$ est la deuxième plus petite valeur propre du Laplacien du graphe.

Idée de la preuve : Soit S l'ensemble des agents avec le max, augmenter la taille de cet ensemble se fait par une communication à travers la frontière de cet ensemble. Or, l'inégalité $\frac{|\partial S|}{|S|} \geq \lambda_2^{\mathbf{L}}\left(1-\frac{|S|}{N}\right)$ lie le nombre de liens qui traversent la frontière de S avec le $\lambda_2^{\mathbf{L}}$ du graphe. Cette inégalité permet d'obtenir une borne sur la probabilité d'augmenter la taille de S et donc sur l'espérance du temps pour augmenter le nombre de capteurs informés.

Théorème. *Pour le* RANDOM-BROADCAST-MAX*, on a*

$$\begin{aligned} \mathbb{E}[\tau] &\leq N\Delta_{\mathscr{G}} + N(\Delta_{\mathscr{G}}-1)\log\left(\tfrac{N-1}{\Delta_{\mathscr{G}}-1}\right) \triangleq \beta \\ and\ \tau &\leq \beta + N\Delta_{\mathscr{G}}\left(\log\left(\tfrac{\Delta_{\mathscr{G}}}{\varepsilon}\right)-1\right) \text{ avec probabilité } 1-\varepsilon \end{aligned}$$

où $\Delta_{\mathscr{G}}$ est le diamètre du graphe.

Idée de la preuve : nous calculons la probabilité que le capteur ayant initialement le max se réveille, puis la probabilité que les voisins de celui-ci se réveillent, puis les voisins des voisins, etc. Le nombre maximal d'itérations étant $\Delta_{\mathscr{G}}$.

Des considérations sur la finesse de ces bornes et la proximité avec des résultats de la littérature du *rumor spreading* peuvent être trouvées dans la Section 2.4 du manuscrit principal.

III CALCUL DISTRIBUÉ DE LA MOYENNE

Nous étudions ici le problème du calcul distribué de la moyenne. Dans ce problème, le but est que les agents du réseau connaissent la moyenne des valeurs initiales des agents en utilisant des communications locales et asynchrones.

III.1 Introduction

Le problème du calcul distribué de la moyenne est un des plus étudiés dans la littérature des réseaux de capteurs sans-fil. Le papier fondateur de Boyd *et al.*[11] introduisait un algorithme asynchrone très simple résolvant ce problème : le *Random Gossip*. Dans cet algorithme, à chaque itération, un agent tiré aléatoirement choisit un de ses voisins de manière aléatoire, ils échangent ensuite leurs variables et se mettent à jour en prenant la moyenne entre leur ancienne valeur et la valeur reçue. Les améliorations algorithmiques proposées ensuite par la littérature sont principalement de deux natures : i) une meilleure exploitation de la géométrie du réseau; ou ii) l'utilisation de communications de type *broadcast*.

La structure du réseau a en effet été exploitée par [37, 38] afin d'obtenir un meilleur mélange à travers le graphe. Pour cela, [37] proposa un algorithme proche du *Random Gossip* mais dans lequel les deux nœuds mis en œuvre ne sont pas nécessairement connectés ; tandis que [38] conçut un algorithme basé sur un moyennage suivant un aller-retour sur un chemin aléatoire. Ces méthodes accélèrent la vitesse de convergence mais demandent un routage important et l'existence impérative d'une voie de retour, ce qui peut être restrictif dans le cas de réseaux sans-fil (si les agents sont mobiles par exemple). L'utilisation de communications de type *broadcast* semble une voie prometteuse à l'aune des résultats de la partie précédente. Le premier algorithme de ce type a été développé par [7] et s'appelle *Broadcast Gossip*, dans celui-ci un agent choisi aléatoirement *broadcast* sa valeur à tous ses voisins qui se mettent à jour en prenant la moyenne de leurs valeurs précédente et reçue. Cet algorithme très simple et sans voie de retour converge très rapidement mais la valeur du consensus n'est pas la moyenne cherchée. Des algorithmes convergents utilisant des communications *broadcast* ont ensuite été proposés ([39, 40]) cependant ceux-ci ont besoin de voies de retour avec tous leurs voisins ce qui peut être problématique dans notre cadre. Finalement, trouver un algorithme asynchrone de calcul distribué de la moyenne basé sur des communications *broadcast* sans voie de retour est toujours une question ouverte que nous nous proposons de résoudre.

Cependant, le simple modèle d'agents avec une seule variable, effectuant des combinaisons linéaires n'est pas adapté au calcul de la moyenne avec des communications sans voies de retour (cette affirmation sera justifiée plus tard). Un autre formalisme appelé *Sum-Weight* introduit par Kempe en 2003 [43] s'est montré particulièrement efficace pour la conception d'algorithmes de moyennage sans voie de retour [42]. Dans ce formalisme, chaque agent maintient deux variables, une initialisée avec sa valeur initiale, l'autre à 1. Les deux variables sont alors mises à jour simultanément à chaque itération et l'estimée de la moyenne par l'agent

est le quotient de ses deux variables. Ce formalisme a été étudié en détail dans le cas synchrone par [43], puis sa convergence a été prouvée dans le cas asychrone par [42, 41]. En revanche, la vitesse de convergence n'a jamais été analysée théoriquement dans le cas asynchrone.

Les contributions de cette partie sont donc doubles : i) l'analyse théorique de la vitesse de convergence des algorithmes de moyennage asynchrones basés sur le formalisme *Sum-Weight* et ii) la conception d'un algorithme basé sur des communications *broadcast* sans voie de retour qui converge plus rapidement que les algorithmes existants.

III.2 Préliminaires sur les Matrices Positives

Nous rappelons ici quelques résultats importants principalement basés sur [3, Chapitres 6.2 et 8].

Soit $\mathbf{A}$ une matrice carrée de taille N, nous notons $\mathbf{A} \geq 0$ ($\mathbf{A}$ est *positive*) si $\forall i, j$, $\mathbf{A}_{i,j} \geq 0$. De même, nous notons $\mathbf{A} > 0$ ($\mathbf{A}$ est *strictement positive*) si $\forall i, j$, $\mathbf{A}_{i,j} > 0$. Ensuite, pour toute matrice positive $\mathbf{A}$, nous notons $\mathcal{G}(\mathbf{A})$ le *graphe induit* par $\mathbf{A}$ c'est-à-dire le graphe de N nœuds tel qu'il y a un lien de i à j de poids $\mathbf{A}_{i,j}$ si et seulement si $\mathbf{A}_{i,j} > 0$ ($\mathbf{A}$ est donc la matrice d'adjacence de $\mathcal{G}(\mathbf{A})$). Par simple calcul, nous voyons qu'il y a un chemin de i à j de longueur m si et seulement si $(\mathbf{A}^m)_{i,j} > 0$, nous allons donc nous intéresser aux puissances des matrices positives.

Soit $\mathbf{A}$ une matrice positive, $(\mathbf{I} + \mathbf{A})^{N-1} > 0$ si et seulement si $\mathbf{A}$ est *irréductible*. Comme $(\mathbf{I} + \mathbf{A})^{N-1} = \sum_{i=0}^{N-1} \binom{N-1}{i} \mathbf{A}^i$, l'irréductibilité de $\mathbf{A}$ implique que pour tout i, j il y a au moins une des matrices $\mathbf{A}, \mathbf{A}^2, ..., \mathbf{A}^{N-1}$ qui a un coefficient (i, j) positif, ce qui veut dire qu'il y a un chemin entre i et j ; en d'autres termes, $\mathcal{G}(\mathbf{A})$ est fortement connecté.

Soit $\mathbf{A}$ une matrice positive, $\exists m$ tel que $\mathbf{A}^m > 0$ si et seulement si $\mathbf{A}$ est *primitive*. De manière similaire au cas irréductible, $\mathcal{G}(\mathbf{A})$ est fortement connecté. La primitivité de $\mathbf{A}$ implique également qu'il existe $m > 0$ tel que pour tout $m' \geq m$ et $i, j \in V$ il y existe un chemin de longueur m' entre i et j. Notons finalement que si un graphe a une matrice d'adjacence primitive, il suffit d'ajouter une lien entre un nœud et lui-même pour rendre cette matrice primitive. Enfin, le théorème de Perron-Frobenius donne les propriétés spectrales particulières des matrices primitives.

Théorème (Théorème de Perron-Frobenius pour les matrices primitives)**.** *Soit* $\mathbf{A}$ *une matrice primitive. Alors*

i) Le rayon spectral $\rho(\mathbf{A})$ *est valeur propre simple de* $\mathbf{A}$*;*
ii) Il y a un vecteur à éléments positifs x *tel que* $\mathbf{A}x = \rho(\mathbf{A})x$*;*
iii) $\rho(\mathbf{A})$ *est la seule valeur propre de module maximal.*

Une autre propriété, très différente, qui nous intéressera par la suite est la *stochasticité* des matrices. Une matrice positive $\mathbf{A}$ est dite *stochastique (par ligne)* quand les sommes de chacune des lignes sont égales à 1 ($\mathbf{A}\mathbb{1} = \mathbb{1}$). Une matrice positive $\mathbf{A}$ est dite *stochastique par colonne* quand les sommes de chacune des colonnes sont égales à 1 ($\mathbb{1}^{\mathrm{T}}\mathbf{A} = \mathbb{1}^{\mathrm{T}}$). Enfin, une matrice stochastique par ligne et par colonne est dite *doublement stochastique*. Il est intéressant de

remarquer que dès qu'une matrice est stochastique par ligne ou par colonne, son rayon spectral est égal 1 et lié au vecteur propre (à droite ou à gauche selon qu'elle soit stochastique par ligne ou par colonne) $\mathbb{1}$ [3, Lemme 8.1.21].

Finalement, considérons une matrice $\mathbf{W}$ *stochastique (par ligne) et primitive*. Le rayon spectral 1 est valeur propre et $\mathbb{1}$ est le vecteur (dit de Perron) associé, cette valeur propre est de module maximal donc toute autre valeur propre a un module inférieur à 1 ; ainsi en multipliant $\mathbf{W}$ par elle-même, cet espace propre va rester stable alors que l'espace orthogonal va décroître exponentiellement. Mathématiquement, $\mathbf{W}^k \longrightarrow_{k\to\infty} \mathbb{1}v^{\mathrm{T}}$ les puissances de cette matrice vont donc tendre vers une matrice de rang 1 égale à $\mathbb{1}v^{\mathrm{T}}$ où v est le vecteur propre à gauche lié à la valeur propre 1. Si cette matrice représentait les échanges entre les agents à chaque instant,

$$x^k = \mathbf{W}^k x^0 \longrightarrow_{k\to\infty} \mathbb{1}v^{\mathrm{T}} x^0$$

donc les variables des agents tendraient toutes vers la même valeur $v^{\mathrm{T}}x^0$. Les matrices primitives et stochastiques jouent donc un grand rôle dans les algorithmes de consensus et plus particulièrement dans le calcul distribué de la moyenne.

III.3 Algorithmes standards de moyennage (à une variable)

Comme précisé dans la première partie, nous considérons un réseau d'agents sans-fil modélisé par un graphe $\mathcal{G} = (V, E)$ et chaque agent i possède une valeur initiale x_i^0. Le but des algorithmes distribués de moyennage est que chaque capteur connaisse $x_{\text{ave}} \triangleq 1/N \sum_{i=1}^N x_i^0$ à l'aide de communications locales asynchrones et de moyennages (éventuellement pondérés) entre les valeurs propres et reçues. Ainsi, dans les algorithmes standards une itération peut être écrite de manière matricielle par :

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} x^k = \mathbf{K}_{\xi^{k+1}} \ldots \mathbf{K}_{\xi^1} x^0$$

où le processus $\{\mathbf{K}_{\xi^k}\}_{k>0}$ à valeurs dans un ensemble $\mathcal{K} = \{\mathbf{K}_m\}_{m=1,\ldots,M}$ est indépendent et identiquement distribué (i.i.d.).

Intéressons nous aux propriétés de ces matrices de mise à jour. Comme les agents effectuent des moyennes pondérées de leurs valeurs reçues et précédente, ces matrices sont nécessairement stochastiques. En effet, pour tout $i \in V$, $x_i^{k+1} = \sum_j (\mathbf{K}_{\xi^{k+1}})_{i,j} x_j^k$ et $\sum_j (\mathbf{K}_{\xi^{k+1}})_{i,j} = 1$ car l'agent effectue une moyenne pondérée. De plus, si les liens existant dans le graphe sous-jacent sont tous utilisés, $\sum_m \mathbf{K}_m$ (et donc $\mathbb{E}[\mathbf{K}]$) a des coefficients non-nuls aux mêmes endroits que $\mathbf{A}$ ainsi que sur la diagonale si les agents gardent une partie de leur information. Donc, si tous les liens possibles sont utilisés et que les capteurs effectuent une moyenne pondérée de leurs valeurs reçues et de leur ancienne valeur avec des coefficients non-nuls, alors les matrices de $\mathcal{K}$ sont stochastiques et $\mathbb{E}[\mathbf{K}]$ est primitive. Nous avons vu dans la section précédente l'importance des propriétés de stochasticité et de primitivité dans l'étude de tels algorithmes. Les matrices étudiées ne sont désormais plus fixes mais aléatoires, le résultat de convergence vers une matrice de rang 1 ne tient plus mais un résultat très similaire existe appelé *ergodicité forte*.

Lemme. *Si chaque matrice de $\mathscr{K}$ est stochastique par ligne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive, alors il existe un vecteur aléatoire à éléments positifs, de somme unité, v tel que*

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} \ldots \mathbf{K}_{\xi^1} x^0 \longrightarrow_{k\to\infty} \mathbb{1} v^{\mathrm{T}} x^0 \quad \textit{presque surement.}$$

De plus, si $\mathbb{E}[K]$ est stochastique par colonne, alors $\mathbb{E}[v^{\mathrm{T}} x^0] = x_{ave}$ donc l'estimateur est non-biaisé.

Exemple : *Broadcast Gossip*. L'algorithme *Broadcast Gossip* [7], repose à chaque itération sur un capteur choisi aléatoirement avec une loi uniforme qui *broadcast* sa valeur à tous ses voisins. Ceux-ci remplacent alors leur valeur par la moyenne de leurs valeurs reçues et précédente. Cet algorithme est basé sur des communications *broadcast* sans voie de retour comme voulu.

Broadcast Gossip

Soit i l'agent actif au temps k.

- i *broadcast* sa valeur à tous ses voisins.
- Chaque voisin $j \in \mathscr{N}_i$ se met à jour : $x_j^{k+1} = \frac{x_i^k + x_j^k}{2}$.

Matriciellement, les matrices de mise à jour sont de la forme (5 envoie à ses voisins 2, 3 et 4)

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Cet algorithme vérifie donc les conditions du lemme précédent, il converge presque sûrement vers un consensus autour de la valeur $v^{\mathrm{T}} x^0$, différente de x_{ave} avec probabilité 1. On peut aussi remarquer que $\mathbb{E}[K]$ est stochastique par colonne, l'algorithme est donc non-biaisé.

Example: *Random Gossip*. L'algorithme *Random Gossip* [11], repose à chaque itération sur un échange pair à pair entre un capteur choisi aléatoirement avec une loi uniforme et un de ses voisins choisi uniformément. Ces deux agents calculent ensuite la moyenne de leurs deux valeurs.

Random Gossip

Soit i l'agent actif au temps k.

- i choisit un voisin j uniformément dans $\mathscr{N}_i$ et ils échangent leurs valeurs.
- i et j se mettent à jour : $x_i^{k+1} = x_j^{k+1} = \frac{x_i^k + x_j^k}{2}$.

Matriciellement, les matrices de mise à jour sont de la forme (ici 1 et 2 échangent)

$$\begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Comme précédemment, cet algorithme vérifie les conditions du lemme précédent et donc il converge presque sûrement vers un consensus autour de la valeur $v^{\mathrm{T}}x^0$. Cependant les matrices de mise à jour de cet algorithme ont une propriété supplémentaire : elles sont également stochastiques par colonne. Ce faisant, $\mathbb{1}^{\mathrm{T}} = \mathbb{1}^{\mathrm{T}}\mathbf{K}_{\xi^{k+1}}...\mathbf{K}_{\xi^1} \longrightarrow_{k\to\infty} \mathbb{1}^{\mathrm{T}}\mathbb{1}v^{\mathrm{T}} = Nv^{\mathrm{T}}$ donc $v = 1/N\mathbb{1}$ presque sûrement. Ainsi, le *Random Gossip* converge vers un consensus autour de $1/N\mathbb{1}^{\mathrm{T}}x^0 = x_{\mathrm{ave}}$ presque sûrement.

Conclusion: la primitivité de l'espérance des matrices de passage semble être une condition légère assurant une diffusion suffisante de l'information dans le réseau. En ce qui concerne le côté stochastique des matrices de mise à jour :

- la *stochasticité par ligne* permet d'atteindre un consensus stable. Ceci peut se comprendre par le fait que si $x^k = c\mathbb{1}$ alors $x^{k+1} = \mathbf{K}_{\xi^{k+1}}c\mathbb{1} = c\mathbb{1}$.
- la *stochasticité par colonne* permet de conserver l'information voulue dans le réseau. Plus précisément, cette propriété permet de conserver la somme des valeurs initiales $\mathbb{1}x^0 = s$ car $\mathbb{1}^{\mathrm{T}}x^{k+1} = \mathbb{1}^{\mathrm{T}}\mathbf{K}_{\xi^{k+1}}...\mathbf{K}_{\xi^1}x^0 = \mathbb{1}x^0 = s$.

Au final, alors que la stochasticité par ligne semble naturelle, la stochasticité par colonne est obligatoire pour espérer converger vers la moyenne des valeurs initiales du réseau. Ainsi, nous avons vu qu'il était impossible de construire des algorithmes avec des matrices de mise à jour stochastiques par ligne mais pas par colonne. Malheureusement, des matrices de mise à jour doublement stochastiques imposent une voie de retour. En effet, lors qu'il y a *broadcast* sans voie de retour, l'émetteur ne dispose que de sa valeur courante et les récepteurs des valeurs reçue et courante, les coefficients non-nuls sur lesquels l'algorithme peut jouer sont indiqués par un carré ci-dessous. En gras, les coefficients correspondant aux valeurs des récepteurs dans la combinaison linéaire de l'émetteur, ces coefficients doivent impérativement rester nuls.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \square & 0 & 0 & \square \\ 0 & 0 & \square & 0 & \square \\ 0 & 0 & 0 & \square & \square \\ 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \square \end{bmatrix}$$

Dans cette matrice, il est donc possible d'assurer la stochasticité par ligne *ou* par colonne mais pas simultanément (sinon la matrice résultante est l'identité). Comme la stochasticité par ligne est vouée à l'échec comme illustré par l'exemple sur le *Broadcast Gossip*, nous verrons dans la partie suivante comment un nouveau formalisme nous permet de concevoir des

algorithmes basés sur des matrices de mise à jour stochastiques par colonne.

III.4 Algorithmes de moyennage basés sur le *Sum-Weight*

Étudions les produits de matrices stochastiques par colonnes choisies dans un ensemble $\mathscr{K} = \{\mathbf{K}_m\}_{m=1,\dots,M}$ à l'aide d'un processus i.i.d. et dont l'espérance $\mathbb{E}[\mathbf{K}]$ est primitive. Le résultat d'ergodicité forte énoncé dans la section précédente ne tient plus, en revanche un résultat d'*ergodicité faible* existe. Le lemme suivant implique qu'un tel produit de matrice va tendre vers une matrice de rang 1 mais cette matrice n'est pas fixe, contrairement au cas de l'ergodicité forte.

Lemme. *Si chaque matrice de $\mathscr{K}$ est stochastique par colonne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive, alors il existe une séquence aléatoire de vecteurs à éléments positifs, de somme unité, $\{v^k\}_{k>0}$ tel que*

$$x^{k+1} = \mathbf{K}_{\xi^{k+1}} \dots \mathbf{K}_{\xi^1} x^0 \sim_{k\to\infty} v^k \mathbb{1}^{\mathrm{T}} x^0 \quad \textit{presque sûrement.}$$

Ainsi, $x^k \sim_{k\to\infty} v^k N x_{\text{ave}}$ donc la somme des valeurs initiales est bien conservée mais la variable ne tend pas vers un consensus. On peut d'ailleurs remarquer que $N v_i^k$ est le coefficient inconnu qui multiplie x_{ave} pour l'agent i au temps k.

L'idée du *Sum-Weight* [43] est que si on considère deux variables par agent, une appelée *somme*, notée s, initialisée avec les valeurs initiales x^0 ; et une appelée *poids*, notée w, initialisée avec $\mathbb{1}$. Ainsi, l'application du lemme précédent donne avec probabilité 1 que

$$\begin{cases} s^k \sim_{k\to\infty} v^k N x_{\text{ave}} \\ w^k \sim_{k\to\infty} v^k N \end{cases}$$

ainsi le quotient coefficient à coefficient de ces deux vecteurs $x^k \triangleq s^k / w^k$ converge donc presque sûrement vers un consensus autour de la moyenne x_{ave}. Ce résultat ainsi que l'approche par l'ergodicité faible est du à Bénézit [38].

Intéressons nous désormais à une preuve plus analytique qui nous permettra d'analyser en détail ces algorithmes et notamment leur vitesse de convergence. Intuitivement, en notant $\mathbf{J}_\perp \triangleq \mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^{\mathrm{T}}$, nous avons

$$\begin{aligned} x^k &= \frac{\mathbf{K}_{\xi^k}\dots\mathbf{K}_{\xi^1}\frac{1}{N}\mathbb{1}\mathbb{1}^{\mathrm{T}}x^0}{w^k} + \frac{\mathbf{K}_{\xi^k}\dots\mathbf{K}_{\xi^1}\mathbf{J}_\perp x^0}{w^k} \\ &= x_{\text{ave}}\mathbb{1} + \frac{\mathbf{K}_{\xi^k}\dots\mathbf{K}_{\xi^1}\mathbf{J}_\perp x^0}{w^k} \end{aligned}$$

donc prouver la convergence de x^k vers $x_{\text{ave}}\mathbb{1}$ revient à prouver la décroissance de $\mathbf{P}^{1,k}\mathbf{J}^\perp x^0/w^k$ vers 0. Comme il est difficile de travailler avec des quotients de vecteurs, nous bornons ce terme comme suit

$$\|x^k - x_{\text{ave}}\mathbb{1}\|_2^2 \le \Psi_1^k \Psi_2^k \quad \text{with} \quad \begin{aligned} \Psi_1^k &= \frac{\|x^0\|_2^2}{[\min_i w_i^k]^2} \\ \Psi_2^k &= \left\|\mathbf{K}_{\xi^k}\dots\mathbf{K}_{\xi^1}\mathbf{J}_\perp\right\|_{\mathrm{F}}^2 \end{aligned}$$

avec

- Ψ_1^k: terme non linéaire, w_i^k peut devenir très petit
- Ψ_2^k: $\mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1} \sim v^k \mathbb{1}^{\mathrm{T}} \Rightarrow \mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1}\mathbf{J}_\perp \sim v^k \mathbb{1}^{\mathrm{T}}\mathbf{J}_\perp = 0$, caractérise l'écart de $\mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1}$ à une matrice de rang 1, il est raisonnable de penser que ce terme décroît exponentiellement vers 0.

Notre preuve sera donc divisée en deux parties : i) prouver qu'il existe une constante $C < \infty$ telle que l'événement $\{\Psi_1^k \leq C\}$ intervient infiniment souvent ; et ii) $\mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1}\mathbf{J}_\perp$ décroît exponentiellement. *Les détails des preuves des résultats suivants ne sont pas repris dans ce résumé, elles peuvent être trouvées Section 3.4.*

Étude de Ψ_1. Le caractère i.i.d. de la suite de matrices $\{\mathbf{K}_{\xi^k}\}_{k>0}$ et la primitivité de $\mathbb{E}[\mathbf{K}]$ permet de prouver qu'il existe une constante $L < \infty$ telle que $\mathbb{P}[\mathbf{K}_{\xi^L}...\mathbf{K}_{\xi^1} > 0] > O$, c'est à dire que la probabilité que le produit de L matrices de mise à jour consécutives soit une matrice à éléments strictement positifs est elle même strictement positive. Comme i) le plus petit coefficient de ce produit de matrice m_L est strictement positif ; ii) que $w^{k+L} = \mathbf{K}_{\xi^{k+L}}...\mathbf{K}_{\xi^k} w^k$ et iii) que w^k est un vecteur à éléments positifs dont la somme est égale à N ; alors la probabilité que $\{w^{k+L} > m_L \mathbb{1}\}$ est strictement positive. Ainsi, il existe un processus i.i.d. de variables aléatoires distribuées géométriquement (à un facteur multiplicatif L près) $\{\Delta_n\}_{n>0}$ tel que l'événement $\{w^k > m_L \mathbb{1}\}$ est vrai le long d'une séquence de temps d'arrêts $\{\tau_n\}_{n>0} \triangleq \{\sum_{\ell=1}^n \Delta_\ell\}_{n>0}$.

Proposition. *Si chaque matrice de $\mathscr{K}$ est stochastique par colonne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive, alors il existe un processus i.i.d. de variables aléatoires distribuées géométriquement $\{\Delta_n\}_{n>0}$ tel que pour tout $n > 0$,*

$$\Psi_1^{\tau_n} \leq \|x^0\|_2^2 (m_L)^{-2L}$$

avec $\tau_n = \sum_{\ell=1}^n \Delta_\ell$.

Étude de Ψ_2. Ce terme faisant intervenir le produit des matrices de mise à jour multiplié par le projecteur sur l'espace orthogonal au consensus $\mathbb{1}$ est classique dans la littérature sur le moyennage distribué. En revanche, la technique habituelle pour prouver la décroissance exponentielle de son espérance repose sur une récursion tirée d'une inégalité qui n'est pas assez fine dans notre cas. Nous avons donc travaillé sur une matrice plus compliquée basée sur des produits de Kronecker '$\otimes$' proposée dans [6]. Ainsi, $\Psi_2^k = \|\mathbf{\Xi}^k\|_{\mathrm{F}}$ où $\|\cdot\|_{\mathrm{F}}$ et

$$\mathbf{\Xi}^k \triangleq \left(\mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1}\mathbf{J}^\perp\right) \otimes \left(\mathbf{K}_{\xi^k}...\mathbf{K}_{\xi^1}\mathbf{J}^\perp\right).$$

En utilisant les propriétés du produit de Kronecker et l'indépendance des matrices de mise à jour, nous prouvons que

$$\mathbb{E}\left[\mathbf{\Xi}^k\right] = (\mathbf{R})^k$$

avec

$$\mathbf{R} \triangleq \mathbb{E}\left[\mathbf{K} \otimes \mathbf{K}\right] \left(\mathbf{J}^\perp \otimes \mathbf{J}^\perp\right).$$

Il reste donc à prouver que le rayon spectral de $\mathbf{R}$ est strictement inférieur à 1. L'idée de la preuve est basée sur le fait que $\mathbb{E}[\mathbf{K} \otimes \mathbf{K}]$ est primitive et stochastique par colonne si $\mathbb{E}[\mathbf{K}]$

l'est, donc son rayon est 1 et les autres valeurs propres sont strictement inférieures en module (le calcul détaillé de ce rayon spectral peut être trouvé Section 3.4.3). Finalement, nous avons le résultat suivant sur la décroissance exponentielle de l'espérance de Ψ_2.

Proposition. *Si chaque matrice de $\mathscr{K}$ est stochastique par colonne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive,*

$$\mathbb{E}[\Psi_2^k] = \mathscr{O}\left(k^{N-2}\mathrm{e}^{-\omega k}\right)$$

avec $\omega = -\log(\rho(\mathbf{R})) > 0$.

Résultats principaux. A l'aide des propositions ci-dessus, nous sommes désormais capables d'énoncer nos contributions principales. Mais d'abord, remarquons qu'au temps $k+1$, pour tout i,

$$\begin{aligned} x_i^{k+1} &= \frac{\sum_{j=1}^N \mathbf{K}_{i,j} s_j^k}{\sum_{j=1}^N \mathbf{K}_{i,j} w_j^k} = \frac{\sum_{j=1}^N \mathbf{K}_{i,j} w_j^k x_j^k}{\sum_{j=1}^N \mathbf{K}_{i,j} w_j^k} \\ &= \sum_{j=1}^N \left(\frac{\mathbf{K}_{i,j} w_j^k}{\sum_{l=1}^N \mathbf{K}_{i,l} w_l^k} \right) x_j^k \end{aligned}$$

où $\mathbf{K}$ correspond à n'importe quelle matrice de $\mathscr{K}$. Donc x_i^{k+1} est un barycentre des $\{x_j^k\}_{j=1,\dots,N}$. Ainsi, en utilisant le fait que $x_{\text{ave}}\mathbb{1}$ vérifie la même inégalité, nous avons $\forall i = 1,\dots,N$,

$$\begin{aligned} |x_i^{k+1} - x_{\text{ave}}| &\leq \sum_{j=1}^N \left(\frac{\mathbf{K}_{i,j} w_j^k}{\sum_{l=1}^N \mathbf{K}_{i,l} w_l^k} \right) |x_j^k - x_{\text{ave}}| \\ &\leq \max_j |x_j^k - x_{\text{ave}}| \end{aligned}$$

et donc $\|x^{k+1} - x_{\text{ave}}\mathbb{1}\|_\infty \leq \|x^k - x_{\text{ave}}\mathbb{1}\|_\infty$, l'erreur par rapport au consensus sur la moyenne décroît donc en norme infini.

En combinant cette décroissance en norme infini et les deux propositions énoncées plus haut, nous obtenons notre théorème de convergence presque sure sous des conditions nécessaires et suffisantes.

Théorème. *Si chaque matrice de $\mathscr{K}$ est stochastique par colonne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive,*

$\{x^k\}_{k>0}$ est bornée et converge vers le consensus autour de la moyenne $x_{ave}\mathbb{1}$ presque sûrement.

De plus, si $\mathbb{E}[\mathbf{K}]$ n'est pas primitive, $\{x^k\}_{k>0}$ ne converge pas vers le consensus autour de la moyenne pour quelques valeurs de x^0 avec probabilité 1.

Le résultat suivant sur la vitesse de convergence correspond à notre principale contribution sur les algorithmes de moyennage distribués. Pour ce théorème, nous introduisons la notation suivante : pour deux séquences de variables aléatoires $\{X^k\}_{k>0}$ et $\{Y^k\}_{k>0}$, nous dirons que $X^k = o_{\text{a.s.}}(Y^k)$ si $X^k/Y^k \to_{k\to\infty} 0$ presque sûrement.

Théorème. *Si chaque matrice de $\mathscr{K}$ est stochastique par colonne avec une diagonale strictement positive et que $\mathbb{E}[\mathbf{K}]$ est primitive, l'erreur quadratique (*SE*) est bornée par une suite décroissant vers* 0*. De plus, elle est également bornée par une fonction exponentiellement décroissante comme suit :*

$$\mathrm{SE}^{\tau_n} = \|x^{\tau_n} - x_{ave}\mathbb{1}\|_2^2 = o_{a.s.}\left(\tau_n^N \mathrm{e}^{-\omega\tau_n}\right)$$

avec $\omega = -\log(\rho(\mathbf{R})) > 0$ et $\tau_n = \sum_{\ell=1}^{n} \Delta_\ell$ est défini précédemment.

Ce résultat nous dit que l'erreur quadratique décroît exponentiellement le long de temps d'arrêt régulièrement espacés (la différence entre eux suit un processus i.i.d.).

III.5 Algorithme proposé et extensions

Nous venons de prouver la convergence exponentielle des algorithmes de moyennage basés sur le *Sum-Weight* sous des conditions légères. Voyons désormais comment ce formalisme nous permet de concevoir un nouvel algorithme de moyennage distribué basé sur des communications *broadcast* sans voie de retour qui offre des performances remarquables. Nous verrons ensuite quelques extensions innovantes de nos résultats.

L'algorithme *BWGossip*. Comme vu précédemment la propagation d'information est bien plus rapide dans les réseaux sans-fil lorsque l'on utilise des communications de type *broadcast*. Nous avons également montré que les algorithmes standards de moyennage utilisant ces communications souffraient soit de problèmes de convergence soit de la présence d'une voie de retour. Le formalisme *Sum-Weight* nous permet de travailler sur des matrices simplement stochastiques par colonne au lieu de doublement stochastiques, c'est ce qui nous permet de concevoir un algorithme basé sur des communications *broadcast* sans voie de retour.

L'algorithme *Broadcast Weighted Gossip (BWGossip)* est basé sur l'envoi par le capteur s'éveillant de sa paire de variables intelligemment pondérées (afin de garantir la stochasticité par colonne) ; ensuite, les capteurs qui reçoivent ces valeurs les ajoutent simplement à leurs variables courantes. Algorithmiquement, considérons que les agents s'éveillent selon un processus i.i.d.,

BWGossip

Soit i l'agent actif au temps k.
► i *broadcast* une version pondérée de sa paire de valeurs à tous ses voisins $\left(\frac{s_i(t)}{d_i+1}; \frac{w_i(t)}{d_i+1}\right)$
► Chaque voisin $j \in \mathscr{N}_i$ se met à jour :
$$\begin{cases} s_j(t+1) = s_j(t) + \frac{s_i(t)}{d_i+1} \\ w_j(t+1) = w_j(t) + \frac{w_i(t)}{d_i+1} \end{cases}$$
► Le capteur i se met à jour :
$$\begin{cases} s_i(t+1) = \frac{s_i(t)}{d_i+1} \\ w_i(t+1) = \frac{w_i(t)}{d_i+1} \end{cases}$$

Matriciellement, les mises à jour des deux variables ont la forme suivante, qui vérifie les conditions de nos théorèmes (5 envoie a ses trois voisins 2, 3 et 4).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1/4 \\ 0 & 0 & 1 & 0 & 1/4 \\ 0 & 0 & 0 & 1 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 \end{bmatrix}$$

Comparons désormais l'Erreur Quadratique Moyenne (EQM) d'algorithmes de moyennage sur des *Random Geometrical Graphs (RGGs)*[3] de 100 nœuds avec un rayon de base égal à 2.

D'abord, comparons avec la Figure III.1 différents algorithmes de moyennage de la littérature avec le *BWGossip*. En plus de celui-ci, nous simulons ainsi le *Random Gossip* et le *Broadcast Gossip* introduits précédemment ainsi qu'un algorithme proche du *BWGossip* créé par Franceschelli *et al.*[51]. Nous remarquons que le *BWGossip* offre une convergence bien plus rapide que les autres algorithmes de la littérature.

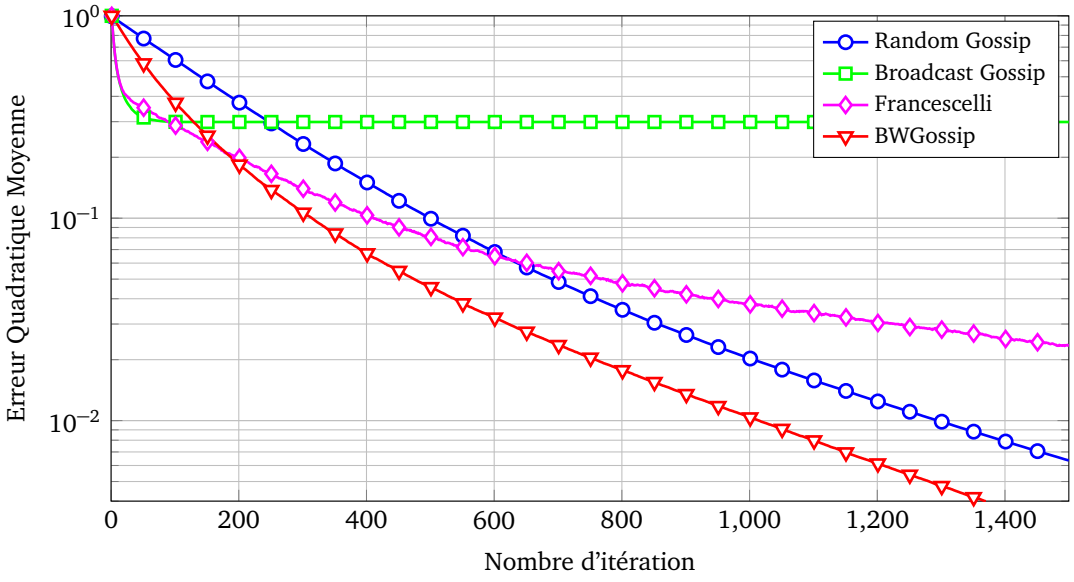


Figure III.1 : EQM du *BWGossip* et d'autres algorithmes de la littérature en fonction du temps.

La Figure III.2 montre la pente du logarithme de l'EQM du *BWGossip* et la borne du théorème précédent ω pour différents nombres d'agents. Nous remarquons que notre borne est très fine (nous traçons la pente en valeur absolue, notre borne supérieure est donc bien sous la courbe empirique).

Extensions de nos résultats. Les résultats présentés dans la section précédente peuvent s'étendre à plusieurs problèmes proches.

Premièrement, il est possible de procéder à une estimation de la somme à la place (ou même en parallèle) de la moyenne en initialisant la variable de poids différemment : il faut

[3]Ces graphes sont obtenus en répartissant uniformément les agents dans $[0,1] \times [0,1]$ et en connectant les agents séparés d'une distance $r_0\sqrt{\log N/N}$ où N est le nombre de nœuds et r_0 le rayon de base

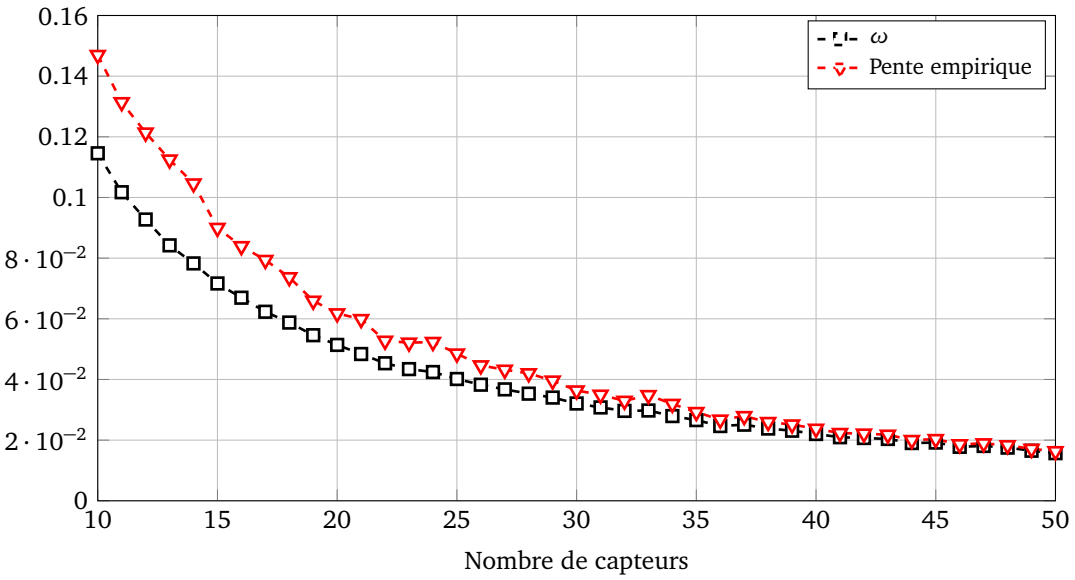


Figure III.2 : Pente empirique du logarithme de l'EQM du *BWGossip* et borne du théorème précedent ω.

qu'un seul capteur initialise sa variable à 1 et tous les autres à 0. Cette initialisation est évidemment problématique, cela peut néanmoins se faire de manière synchrone avec un agent initiateur du calcul. Par ailleurs, le quotient n'est pas défini tant qu'un capteur n'a pas reçu d'information, l'algorithme requiert donc un temps d'initialisation.

Il est également possible de prendre en compte des défaillances i.i.d. de certains liens du graphe tant que i) les agents sont au courant de ces défaillances (par un canal de contrôle par exemple) et ii) que le graphe reste fortement connexe en moyenne. Notre théorème permet même de prévoir la vitesse de convergence dans le cas de telles défaillances en ajoutant les matrices de mise à jour lors de défaillances dans l'ensemble $\mathscr{K}$ et en calculant le rayon spectral de la nouvelle matrice $\mathbf{R}$ (Voir les simulations de la Section 3.5.5 du manuscrit principal).

Finalement, remarquons que lorsque les matrices de mise à jour d'un algorithme *Sum-Weight* sont doublement stochastiques, alors $w^k = \mathbb{1}$ pour tout k, donc $x^k = s^k$, l'algorithme repose donc sur une variable : c'est un algorithme standard. Donc, l'erreur quadratique est donnée par notre deuxième proposition, notre borne ω sur la pente du logarithme de l'EQM est donc valable pour les algorithmes standards de moyennage distribué. Nous traçons dans la Figure III.3, l'EQM empirique du *Random Gossip*, la borne la plus fine de [6] et notre borne ω. Nous remarquons que notre borne est bien meilleure que celle originalement proposée par Boyd *et al.*[6].

III.6 Application à la radio cognitive

Dans un réseau de radio cognitive, des utilisateurs secondaires (SU) ont l'autorisation d'utiliser le medium (généralement la bande de fréquences) lorsque les utilisateurs primaires (PU) ne l'utilisent pas. Afin de savoir s'ils peuvent ou non émettre, les utilisateurs secondaires

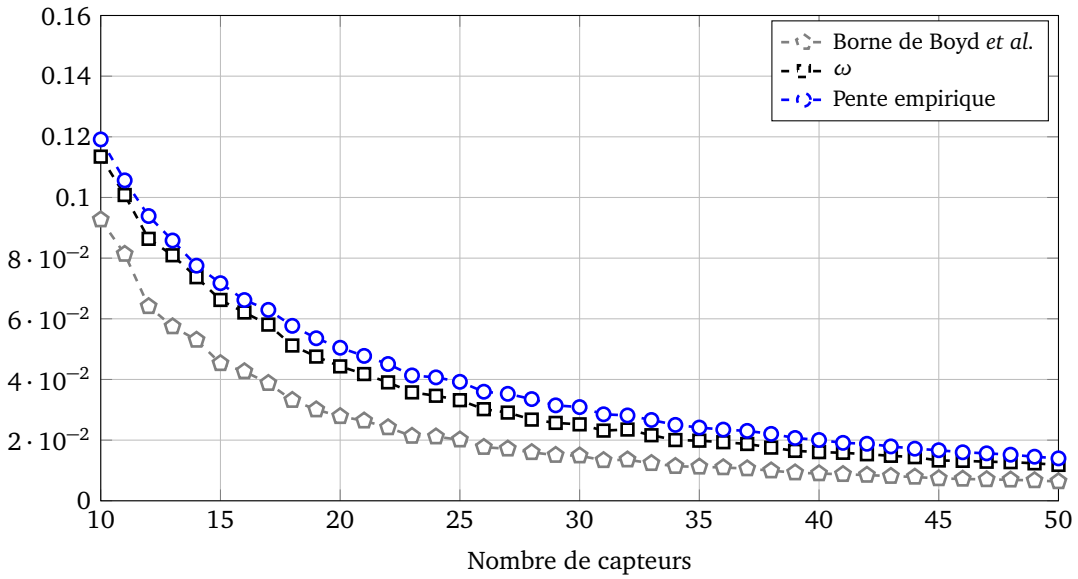


Figure III.3 : Pente empirique du logarithme de l'EQM du *Random Gossip* et bornes associées.

doivent donc sonder la présence de communications sur le medium. Du fait de leur localisations différentes, les utilisateurs secondaires reçoivent plus ou moins bien les émissions des utilisateurs primaires. Afin d'avoir une détection d'émission avec une probabilité de détection cible $P_{\mathrm{D}}^{\mathrm{target}}$ (typiquement donnée par le standard) et une faible probabilité de fausse alarme, le tout en un temps assez court devant la taille d'une trame, les utilisateurs secondaires ont tout intérêt à coopérer, en utilisant par exemple un autre canal dédié.

La manière traditionnelle de gérer ce problème de détection distribuée est de faire parvenir les informations (données ou tests) à un centre de fusion (voir [52] et références). Quelques algorithmes de la littérature sont néanmoins des algorithmes totalement distribués (sans centre de fusion) [53, 54, 55] ; cependant, les seuils utilisés pour la décision sont soit déterminés de manière asymptotique (et donc les probabilités de détections ne sont pas assurées en temps fini) soit choisis de manière centralisée. Distribuer l'étape de calcul du seuil est donc un problème ouvert.

Considérons un réseau de N utilisateurs secondaires cherchant à détecter un signal x dans du bruit n à partir de N_s échantillons. Nous faisons l'hypothèse que s'il existe, le signal x_i^k reçu à l'instant k par le capteur i est gaussien centré et de variance γ_i (le cas où le signal est connu par un séquence d'apprentissage est traité dans le manuscrit à la Section 3.7) ; le bruit n_i^k à l'instant k au capteur i est également gaussien centré et de variance σ_i. Nous considérerons que ces statistiques sont connues par l'agent i. L'hypothèse à tester est donc

$$\begin{cases} \mathscr{H}_0 : & \forall i, \quad y_i = n_i \\ \mathscr{H}_1 : & \forall i, \quad y_i = x_i + n_i \end{cases}$$

et en définissant le Rapport Signal à Bruit au capteur i (RSB_i), le test de Neyman-Pearson

centralisé s'écrit

$$\mathrm{T}(y) \triangleq \frac{1}{N}\sum_{i=1}^{N} \frac{\|y_i\|_2^2}{\gamma_i^2+\sigma_i^2}\mathrm{RSB}_i \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta$$

où η est le seuil de détection choisi pour que la probabilité de bonne détection soit égale à la probabilité de détection cible $\mathbb{P}[\mathrm{T}(y) > \eta | \mathscr{H}_1] = P_{\mathrm{D}}^{\text{target}}$. La distribution de $\mathrm{T}(y)$ peut être approximée par une loi Gamma $\Gamma(\kappa_T, \theta_T)$ avec

$$\kappa_{\mathrm{T}} = \frac{NN_s}{2} \cdot \frac{\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2} \text{ et } \theta_{\mathrm{T}} = \frac{2}{N} \cdot \frac{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i^2}{\frac{1}{N}\sum_{i=1}^{N}\mathrm{SNR}_i}.$$

On en déduit donc que le seuil optimal η donné par la probabilité de détection cible $P_{\mathrm{D}}^{\text{target}}$ vérifie

$$P_{\mathrm{D}}^{\text{target}} = \mathbb{P}[\mathrm{T}(y) > \eta | \mathscr{H}_1] \approx 1 - G_{\kappa_{\mathrm{T}},\theta_{\mathrm{T}}}(\eta)$$

ce qui implique que

$$\eta = G^{(-1)}_{\kappa_{\mathrm{T}},\theta_{\mathrm{T}}}(1 - P_{\mathrm{D}}^{\text{target}})$$

où $G_{\kappa,\theta}$ est la fonction de répartition d'une loi Gamma avec les paramètres (κ, θ) et $G^{(-1)}_{\kappa,\theta}$ est son inverse.

Nous voyons que le test et les paramètres permettant de calculer le seuil font intervenir beaucoup de moyennes, il est donc intéressant de remplacer ces moyennes exactes par N_g itérations d'algorithmes de moyennage distribué présentés plus haut. Notre test distribué

$$T_i(y) = \sum_{j=1}^{N} \mathbf{P}_{i,j}^{1,N_g} \frac{\|y_j\|_2^2}{\gamma_j^2+\sigma_j^2}\mathrm{SNR}_j \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \eta_i,$$

à l'agent i, où $\mathbf{P}^{1,N_g}$ représente la concaténation de N_g étapes de moyennage distribué et η_i est le seuil que doit calculer i.

$$\eta_i = G^{(-1)}_{\kappa_i,\theta_i}\left(1 - P_{\mathrm{D}}^{\text{target}}\right)$$

$$\text{avec } \kappa_i = \frac{N_s}{2} \cdot \frac{\left(\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j\right)^2}{\sum_{j=1}^{N}\left(\mathbf{P}_{i,j}^{1,N_g}\right)^2\mathrm{RSB}_j^2} \text{ et } \theta_i = 2 \cdot \frac{\sum_{j=1}^{N}\left(\mathbf{P}_{i,j}^{1,N_g}\right)^2\mathrm{RSB}_j^2}{\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j}.$$

Malheureusement, $\left(\mathbf{P}_{i,j}^{1,N_g}\right)^2$ ne peut pas s'obtenir de manière distribuée. Une *première approche* consiste à dire que $\left(\mathbf{P}_{i,j}^{1,N_g}\right)^2 \approx 1/N\mathbf{P}_{i,j}^{1,N_g}$, donnant les paramètres suivants

$$\kappa_i^{(1)} = \frac{NN_s}{2} \cdot \frac{\left(\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j\right)^2}{\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j^2} \text{ et } \theta_i^{(1)} = \frac{2}{N} \cdot \frac{\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j^2}{\sum_{j=1}^{N}\mathbf{P}_{i,j}^{1,N_g}\mathrm{RSB}_j}.$$

Ces paramètres sont calculables de manière distribuée, et on peut avoir ainsi une approximation du seuil de manière distribuée. Cependant la connaissance du nombre de capteurs N

est requise. Ceci peut être remplacé dans une *deuxième approche* par le fait de calculer conjointement la somme et la moyenne comme expliqué précédemment avec un algorithme de type *Sum-Weight*. En notant $\mathbf{S}^{1,N_g}$ la concaténation de N_g étapes d'estimation de la somme,

$$\kappa_i^{(2)} = \frac{N_s}{2} \cdot \frac{\left(\sum_{j=1}^{N} \mathbf{S}_{i,j}^{1,N_g} \mathrm{RSB}_j\right)^2}{\sum_{j=1}^{N} \mathbf{S}_{i,j}^{1,N_g} \mathrm{RSB}_j^2} \quad \text{et} \quad \theta_i^{(2)} = 2 \cdot \frac{\sum_{j=1}^{N} \mathbf{M}_{i,j}^{1,N_g} \mathrm{RSB}_j^2}{\sum_{j=1}^{N} \mathbf{S}_{i,j}^{1,N_g} \mathrm{RSB}_j}.$$

Cette approche est, elle, complètement distribuée. Nous traçons dans la Figure III.4 la probabilité de détection et de fausse alarme pour un réseau de 10 agents avec des RSB distribués exponentiellement autour d'un SNR moyen. Nous prenons $N_s = N_g = 64$ et $P_{\mathrm{D}}^{\mathrm{target}} = 0.99$. Les algorithmes considérés sont le détecteur centralisé, le détecteur basé sur l'*approche 1* avec le *Random Gossip* comme algorithme de moyennage et le détecteur basé sur l'*approche 2* avec le *BWGossip* comme algorithme de moyennage/sommation. Nous remarquons que le détecteur basé sur l'*approche 1* a une faible probabilité de fausse alarme mais sa probabilité de détection cible n'est pas atteinte. En revanche, dans le cas de l'*approche 2* avec le *BWGossip*, la perte en probabilité de fausse alarme est relativement petite alors que la probabilité de détection est plus élevée que la probabilité de détection cible, il a donc de très bonnes performances. Une simulation d'un cas de résolution du problème du terminal caché est présenté à la fin de la Section 3.7.

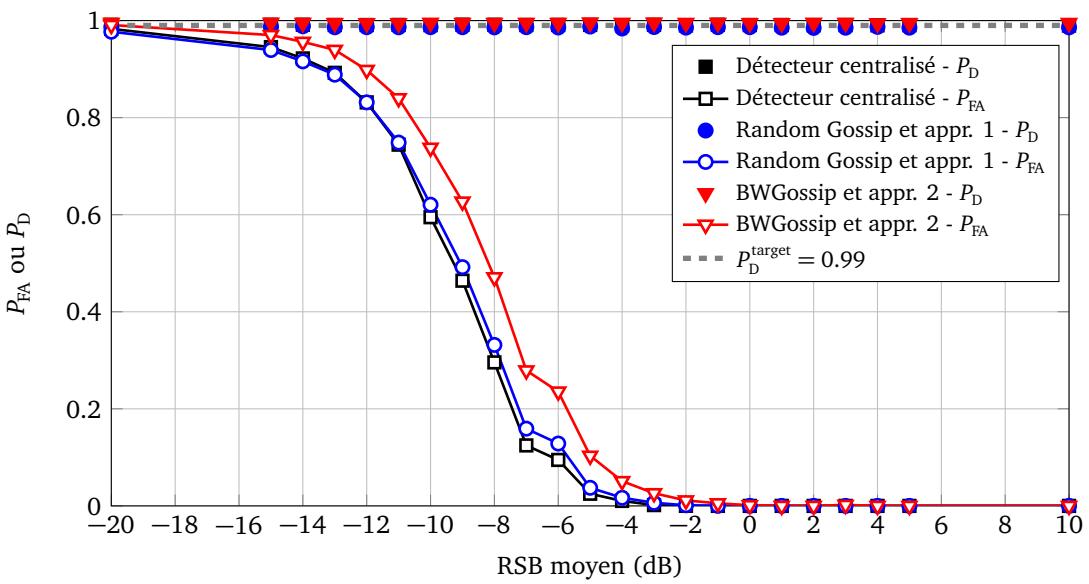


Figure III.4 : P_{FA} et P_{D} en fonction du RSB moyen.

IV Optimisation distribuée

Nous nous intéressons maintenant à l'optimisation distribuée. Après avoir motivé nos choix, nous développons et prouvons la convergence d'un algorithme asynchrone d'optimisation convexe distribuée.

IV.1 Introduction

Les dernières années ont vu une explosion de la quantité de données disponibles sur le Web et dans les capacités calculatoires des équipements électroniques. Le modèle d'inférence classique où une seule machine doit inférer des comportements à partir d'un petit échantillon de données n'est donc pas adéquat.

Afin de prendre en compte de manière efficace ce nouvel environnement [58], il est intéressant de considérer un système où une grande quantité de données est répartie sur de nombreuses machines (physiquement) distantes. Ces machines possèdent de bonnes capacités de calcul, elles ont en revanche besoin de communiquer entre elles car leurs données (même si elles sont importantes) ne sont pas forcément représentatives ou suffisantes (si les données des machines sont de nature différente ou si le système a besoin d'une grande précision). Pour des raisons évidentes de congestion du réseau, des communications locales sont à privilégier par rapport à un système centralisé avec un centre de fusion.

Une large majorité des algorithmes de traitement automatique de données repose sur la minimisation d'une fonction convexe bien choisie et dépendante des données. Pour effectuer un traitement automatique performant sur des réseaux *big data*, il faut donc s'intéresser aux algorithmes d'optimisation distribuée. Mathématiquement, un réseau d'agents représentés par son graphe $\mathcal{G} = (V, E)$ cherche à résoudre de manière distribuée :

$$\min_{x \in \mathbb{R}} \quad f(x) \triangleq \sum_{i \in V} f_i(x)$$

où f_i est une fonction convexe connue seulement par l'agent i.

f n'est disponible nulle part dans le réseau. Pour parvenir à l'optimum voulu, les agents doivent donc i) *mettre à jour* leurs variables locales en utilisant leur fonction privée et ii) *communiquer* leurs variables (la fonction n'est pas échangeable car les données sur lesquelles elle repose sont trop importantes).

Il existe dans la littérature deux grandes classes d'algorithmes résolvant ce problème, elles diffèrent dans la manière qu'ont les agents d'utiliser leur fonctions locales. Dans la première classe, l'agent i utilise des propriétés *locales* de sa fonction f_i telles que le gradient ou le sous-gradient ainsi qu'une étape de moyennage distribuée (vu dans la partie III) [59, 60, 61, 62, 63, 64, 67, 65, 66]. Ces méthodes ont le mérite d'être simples et de se transposer facilement dans un contexte asynchrone (en passant d'un moyennage synchrone à un moyennage asynchrone) cependant elles sont relativement lentes, en particulier dans le cas de fonctions quadratiques (une étude plus détaillée de ces fonctions se trouve Section 4.2). La deuxième classe de méthodes fait intervenir les fonctions des agents dans leur globalité

(sous la forme d'un argmin par exemple). La plus populaire de ces méthodes est l'*Alternating Direction Method of Multipliers (ADMM)* [68], elle s'est montrée particulièrement efficace dans le cadre de communications locales basées sur un graphe [69, 70].

Nous allons donc nous concentrer sur la deuxième classe d'algorithmes et sur l'ADMM en particulier. Malheureusement, bien qu'adapté à l'optimisation distribuée sur les graphes, l'ADMM reste un algorithme *synchrone* c'est à dire que tous les agents doivent effectuer leurs calculs de manière synchrone à chaque itération puis échanger leurs résultats de manière synchrone. Cependant, dans de nombreuses situations, les tailles des données des agents ainsi que leurs capacités de calculs diffèrent largement, le synchronisme devient alors un fardeau car la vitesse globale de l'algorithme est liée à la vitesse de l'agent le plus lent. De plus, nous avons évoqué précédemment que le synchronisme dans les communications d'un réseau pouvait impliquer des collisions et un engorgement du réseau.

Dans cette partie nous allons donc présenter une méthode d'optimisation distribuée basée sur l'ADMM puis nous introduirons les opérateurs monotones, un formalisme clair et puissant pour l'optimisation convexe. Ce formalisme nous permettra de concevoir un algorithme asynchrone d'optimisation distribuée basé sur l'ADMM.

IV.2 Optimisation distribuée par l'ADMM

Le problème original $\min_{x\in\mathbb{R}} f(x) \triangleq \sum_{i\in V} f_i(x)$ n'est pas adapté à l'optimisation sur un graphe car i) il ne traduit pas le fait que chaque agent i a seulement accès à sa fonction f_i et ii) le graphe et donc les communications possibles n'apparaissent pas.

Commençons d'abord par donner à chaque capteur sa propre variable x_i, cela revient à passer d'un problème sur $\mathbb{R}$ à un problème sur $\mathbb{R}^N$ avec une contrainte d'égalité mais désormais chaque composante est liée à une fonction :

$$\begin{aligned} \min_{x\in\mathbb{R}^N} \quad & F(x) \triangleq \sum_{i\in V} f_i(x_i) \\ \text{subject to} \quad & x_1 = x_2 = \ldots = x_N \end{aligned}$$

ce qui peut se réécrire à l'aide d'une indicatrice sur l'espace de consensus comme :

$$\min_{x\in\mathbb{R}^N} \quad F(x) + \iota_{\text{Span}(\mathbb{1})}(x)$$

avec ι_C l'indicatrice telle que $\iota_C(x) = 0$ si $x \in C$ et 0 ailleurs. Ce problème dépend maintenant de la somme de deux fonctions dont une dépend des données des agents et l'autre du fait que l'on veut un consensus. Cependant, cette fonction de consensus ne prend pas en compte le graphe.

Pour faire intervenir le graphe, [69] a proposé d'assurer le consensus non pas sur tout les nœuds du graphe mais sur des sous-ensembles se recoupant. A partir du graphe original $\mathcal{G} = (V, E)$, nous construisons L sous-graphes $\{\mathcal{G}_\ell = (V_\ell, E_\ell)\}_{\ell=1,\ldots,L}$ tels que pour tout sous-graphe $\mathcal{G}_\ell$, V_ℓ et E_ℓ sont des sous-ensembles de V et E respectivement, E_ℓ lie seulement des nœuds de V_ℓ, et $\mathcal{G}_\ell$ est connexe. Il est évident qu'assurer le consensus dans tous les sous-graphes implique un consensus général sur le réseau si

- $\cup_{\ell=1}^{L} V_\ell = V$,
- $(V, \cup_{\ell=}^{L} E_\ell)$ est connexe.

Avec chaque sous-graphe $\mathcal{G}_\ell$, nous définissons la matrice $\mathbf{M}_\ell \in \mathbb{R}^{|V_\ell| \times N}$ remplie de zéros sauf dans un coefficient par ligne égal à 1, le coefficient non nul de la i-ième ligne correspond au nœud à la i-ième position dans V. Soit $M = \sum_{\ell=1}^{L} |V_i|$, nous définissons la matrice $\mathbf{M}$ qui transcrira notre partition du graphe comme la matrice de $\mathbb{R}^{M \times N}$ égale à l'empilement en colonne des sous-matrices $\{\mathbf{M}_\ell\}_{\ell=1,...,L}$. Il est aussi utile d'introduire les notations suivantes: soit z un vecteur de taille M, nous divisions ce vecteur en blocs de taille $|V_1|, ..., |V_L|$ tel que $z = (z_{|1}, ..., z_{|L}) \in \mathbb{R}^{|V_1|} \times ... \times \mathbb{R}^{|V_L|}$. De cette manière si $z = \mathbf{M}x$, nous avons pour tout $\ell = 1, ..., L$ $z_{|\ell} = \mathbf{M}_\ell x$. Pour tout $i \in V$ nous définissons aussi σ_i l'ensemble des indices des sous-graphes auxquels i appartient et pour tout $\ell \in \sigma_i$, nous définissons $z_{i,|\ell}$ comme le coefficient de $z_{|l}$ lié au nœud i.

Avec ces conditions et notations, notre problème se réécrit :

$$\begin{aligned} \min_{x \in \mathbb{R}^N, z \in \mathbb{R}^M} \quad & F(x) + G(z) \\ \text{subject to} \quad & \mathbf{M}x = z \end{aligned}$$

avec $F(x) \triangleq \sum_{i=1}^{N} f_i(x_i)$, $G(z) \triangleq \sum_{\ell=1}^{L} \iota_{\mathrm{Sp}(\mathbb{1}_{|V_\ell|})}(z_{|\ell})$ et où $\iota_{\mathrm{Sp}(\mathbb{1}_{|V_\ell|})}$ est la fonction indicatrice de l'espace engendré par $\mathbb{1}_{|V_\ell|}$.

Ce problème est bien séparé en deux fonctions : une dépendant des données telle que les données de l'agent i interviennent seules dans la i-ième composante et une assurant le consensus basée sur des consensus *locaux*. Appliquons désormais l'ADMM sur ce problème bien adapté à l'optimisation distribuée. Pour cela, soit $\rho > 0$, nous introduisons le *Lagrangien augmenté* du problème :

$$\mathcal{L}_\rho(x, y; \lambda) = F(x) + G(z) + \langle \mathbf{M}x - z; \lambda \rangle + \frac{\rho}{2} \|\mathbf{M}x - z\|_2^2$$

l'ADMM consiste alors en une minimisation alternée du Lagrangien augmenté dans les deux variables primales puis d'une montée de gradient dual (des détails sur les méthodes duales dont l'ADMM fait partie et notamment sur leur convergence se trouvent Section 4.3) :

$$x^{k+1} = \operatorname*{argmin}_x \mathcal{L}_\rho(x, z^k; \lambda^k)$$

$$z^{k+1} = \operatorname*{argmin}_z \mathcal{L}_\rho(x^{k+1}, z; \lambda^k)$$

$$\lambda^{k+1} = \lambda^k + \rho(\mathbf{M}x^{k+1} - z^{k+1})$$

En utilsant les définitions de nos fonctions F et G nous obtenons un algorithme synchrone d'optimisation décrit plus bas.

Cet algorithme remplit bien les conditions énoncées précédemment à savoir que chaque agent utilise sa fonction propre de manière complète et que les échanges effectués sont locaux. En revanche, cet algorithme est synchrone ; nous allons d'abord introduire un formalisme nous

Optimisation Distribuée (Synchrone) basée sur l'ADMM

A chaque itération k:

- Chaque agent i effectue une minimisation :

$$x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

- Chaque sous-ensemble d'agents V_ℓ calcule sa moyenne locale :

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|V_\ell|} \sum_{i \in V_\ell} x_i^{k+1}$$

- Chaque agent i se met à jour :

$$\forall \ell \in \sigma_i, \;\; \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

permettant de comprendre intuitivement cet algorithme avant de concevoir un algorithme asynchrone basé sur l'ADMM.

IV.3 Les opérateurs monotones

Un *opérateur* T sur un espace Euclidien $\mathbb{R}^N$ est un mapping :

$$\begin{array}{rccc} \mathsf{T}: & \mathbb{R}^N & \to & 2^{\mathbb{R}^N} \\ & x & \mapsto & \mathsf{T}(x) \subset \mathbb{R}^N. \end{array}$$

Il peut être identifié de manière équivalente à un sous-ensemble de $\mathbb{R}^N \times \mathbb{R}^N$ et nous écrivons $(x, y) \in \mathsf{T}$ quand $y \in \mathsf{T}(x)$. Nous définissons également les *zéros* d'un opérateur comme $\mathbf{zer}\mathsf{T} = \{x : (x, 0) \in \mathsf{T}\} = \mathsf{T}^{-1}(0)$. L'opérateur *identité* est I défini comme $\{(x, x) : x \in \mathbb{R}^N\}$. Nous donnons quelques propriétés simples pour des opérateurs T et U :

- $\forall \rho \in \mathbb{R}$, $\rho\mathsf{T} = \{(x, \rho y) : (x, y) \in \mathsf{T}\}$;
- $\mathsf{T} + \mathsf{U} = \{(x, y + z) : (x, y) \in \mathsf{T}, (x, z) \in \mathsf{U}\}$;
- $\mathsf{T}^{-1} = \{(y, x) : (x, y) \in \mathsf{T}\}$.

Un opérateur T est dit *monotone* si

$$\forall (x, y), (x', y') \in \mathsf{T}, \;\; \langle x - x'; y - y' \rangle \geq 0$$

Les opérateurs monotones jouent un grand rôle dans la théorie de l'optimisation convexe car si h est une fonction convexe, sa sous-différentielle ∂h est un opérateur monotone. Nous nous intéressons donc à trouver des zéros des opérateurs monotones. Pour cela, nous définissons la *résolvente* d'un opérateur T comme l'opérateur $\mathsf{J}_\mathsf{T} \triangleq (\mathsf{I} + \mathsf{T})^{-1}$. Il est facile de remarquer que trouver un zéro d'un opérateur est équivalent à trouver un point fixe de sa résolvente ; il

semble donc logique de chercher un minimum de h en itérant la résolvente de ∂h :

$$\zeta^{k+1} = \mathsf{J}_{\partial h}(\zeta^k)$$

Nous allons de ce fait nous intéresser aux propriétés de contraction des résolventes d'opérateurs monotones. Ces résultats sont principalement tirés de [80, 81].

Lemme. J *est la résolvente d'un opérateur monotone si et seulement si il est* Fermement Non-Expansif (FNE)*:*

$$\forall (x, x'), \quad \langle x - x'; \mathsf{J}(x) - \mathsf{J}(x') \rangle \geq \|\mathsf{J}(x) - \mathsf{J}(x')\|^2.$$

Les résolventes des opérateurs monotones ne sont donc pas des contractions. La Figure IV.1 représente les notions de ferme non-expansivité, de non-expansivité (ou 1-Lipschitz) et de contraction.

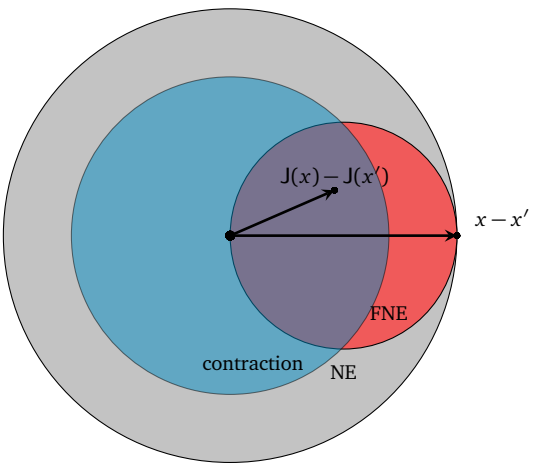


Figure IV.1 : Illustration de la ferme non-expansivité.

Bien que le théorème du point fixe de Banach ne s'applique pas, le résultat suivant affirme la convergence vers un point fixe de l'itération d'un opérateur FNE.

Théorème. *Soit* J *un opérateur FNE tel que* $\mathbf{fix}\,\mathsf{J} \neq \emptyset$*, la suite produite par*

$$\zeta^{k+1} = \mathsf{J}(\zeta^k)$$

converge vers un point de $\mathbf{fix}\,\mathsf{J}$.

L'itération de la résolvente $\mathsf{J}_{\partial h}$ de la sous-différentielle d'une fonction convexe h converge donc vers un minimum de celle-ci. L'algorithme associé est parfois appelé *algorithme du point proximal* :

$$x^{k+1} = \mathsf{J}_{\partial h}(x^k) \Leftrightarrow x^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ h(x) + \frac{1}{2} \left\| x - x^k \right\|^2 \right\}.$$

Retour à notre problème d'optimisation distribuée. Notre problème

$$\begin{aligned} \min_{x \in \mathbb{R}^N, z \in \mathbb{R}^M} \quad & F(x) + G(z) \\ \text{subject to} \quad & \mathbf{M}x = z \end{aligned}$$

est un problème contraint, il est donc usuel d'étudier la *fonction duale*

$$\mathscr{D}(\lambda) \triangleq -F^*(-\mathbf{M}^{\mathrm{T}}\lambda) - G^*(\lambda)$$

où F^* et G^* sont les *transformées de Fenchel-Legendre* de F et G (voir [72, Chapitre 3.3] pour des détails sur la dualité). La fonction duale ne dépend que d'une variable et son maximum (s'il existe) $\lambda^\star$ permet de trouver l'optimum cherché $x^\star$ par le Lagrangien augmenté : $(x^\star, z^\star) = \operatorname{argmin}_{x,z} \mathscr{L}_\rho(x, z; \lambda^\star)$.

Nous voulons donc trouver un zéro de

$$-\partial\mathscr{D} = \underbrace{-\mathbf{M}\partial F^*(-\mathbf{M}^{\mathrm{T}}\cdot)}_{\triangleq \mathsf{U}} + \underbrace{\partial G^*}_{\triangleq \mathsf{V}}$$

où U et V sont des opérateurs monotones. En revanche, comme ils sont de nature très différente, l'un contenant les fonctions des agents et l'autres l'indicatrice du consensus, nous ne voulons pas utiliser $\mathsf{J}_{\mathsf{U}+\mathsf{V}}$ mais les résolventes de U et V séparément.

Le problème, dit de *splitting*, qui consiste à trouver un point fixe de la somme de deux opérateurs, est très étudié. La méthode de *splitting* la plus célèbre, basée sur l'algorithme de Douglas-Rachford [86], consiste à utiliser la résolvente de l'opérateur de Lions-Mercier [87].

Théorème. *La résolvente J^{LM} de l'opérateur de Lions-Mercier de deux opérateurs U et V :*

- *est définie comme $\mathsf{J}^{LM} \triangleq \mathsf{J}_{\rho\mathsf{U}} \circ \big(2\mathsf{J}_{\rho\mathsf{V}} - \mathsf{I}\big) + \big(\mathsf{I} - \mathsf{J}_{\rho\mathsf{V}}\big)$;*
- *est FNE si U et V sont monotones;*
- *a un point fixe si $\mathbf{zer}(\mathsf{U}+\mathsf{V}) \neq \emptyset$.*

Enfin, si $\zeta^\star \in \mathbf{fix}\, \mathsf{J}^{LM}$ alors $\lambda^\star \triangleq \mathsf{J}_{\rho\mathsf{V}}(\zeta^\star) \in \mathbf{zer}(\mathsf{U}+\mathsf{V})$.

Il est possible de montrer (voir Section 4.4.3) que les itérations de la résolvente de l'opérateur de Lions-Mercier avec les opérateurs U et V définis plus haut donnent lieu à l'ADMM. Les itérations de la résolvente se font sur une variable $\zeta \in \mathbb{R}^M$ dont les variables intermédiaires λ, z, x sont tirées (par exemple, $\lambda^k = \mathsf{J}_{\rho\mathsf{V}}(\zeta^k)$). La convergence de l'ADMM n'est alors que l'assemblage du théorème de Lions-Mercier sur le caractère FNE de la résolvente et le théoreme de point fixe des résolventes.

IV.4 Optimisation distribuée asynchrone

Grâce à la théorie des opérateurs monotones, nous savons que l'ADMM consiste en l'itération de la résolvente de Lions-Mercier avec les opérateurs U et V définis plus haut. Il est également possible de remarquer (voir Section 4.4.3) que la mise à jour du ℓ-ième bloc de la variable ζ sur laquelle on itère correspond à la mise à jour des agents appartenant à V_ℓ. Ainsi, si l'on note

$$\zeta^{k+1} = \begin{bmatrix} \zeta_{|1}^{k+1} \\ \vdots \\ \zeta_{|\ell}^{k+1} \\ \vdots \\ \zeta_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathsf{J}_{|1}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \mathsf{J}_{|\ell}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \mathsf{J}_{|L}^{\mathrm{LM}}(\zeta^k) \end{bmatrix} = \mathsf{J}^{\mathrm{LM}}(\zeta^k)$$

mettre à jour uniquement les agents de V_ℓ, correspond à l'itération

$$\zeta^{k+1} = \begin{bmatrix} \zeta^{k+1}_{|1} \\ \vdots \\ \zeta^{k+1}_{|\ell} \\ \vdots \\ \zeta^{k+1}_{|L} \end{bmatrix} = \begin{bmatrix} \zeta^{k}_{|1} \\ \vdots \\ \mathsf{J}^{\mathrm{LM}}_{|\ell}(\zeta^k) \\ \vdots \\ \zeta^{k}_{|L} \end{bmatrix} \triangleq \hat{\mathsf{J}}^{\mathrm{LM}}_{|\ell}(\zeta^k).$$

Il semble donc intéressant d'appliquer la résolvente de Lions-Mercier sur un seul bloc tiré aléatoirement de manière i.i.d. à chaque itération. Cependant, contrairement à J^{LM}, les opérateurs restreints à un bloc $\{\hat{\mathsf{J}}^{\mathrm{LM}}_{|\ell}\}_{\ell=1,\dots,L}$ ne sont pas Fermement Non-Expansifs et donc le théorème de convergence énoncé plus haut ne s'applique pas. Notre contribution majeure dans cette partie fut ainsi de prouver le résultat suivant à propos de la convergence presque sûre des itérations d'une résolvente où seul un bloc choisi aléatoirement est mis à jour (voir Section 4.5 pour la preuve).

Théorème. *Soit $\{\xi^k\}_{k>0}$ un processus i.i.d. à valeurs dans $\{1,\dots,L\}$ tel que $\mathbb{P}[\xi^1 = \ell] > 0 \forall \ell = 1,\dots,L$. Soit J un opérateur FNE tel que $\mathbf{fix}\,\mathsf{J} \neq \emptyset$, la suite produite par*

$$\zeta^{k+1} = \hat{\mathsf{J}}^{LM}_{|\xi^{k+1}}(\zeta^k)$$

converge presque sûrement vers un point de $\mathbf{fix}\,\mathsf{J}$.

Ainsi, la théorie des opérateurs monotones et notre résultat sur les itérations aléatoires des résolventes nous permet, en l'appliquant à la résolvente de Lions-Mercier avec $\mathsf{U} = -\mathbf{M}\partial F^*(-\mathbf{M}^{\mathrm{T}}\cdot)$ et $\mathsf{V} = \partial G^*$, de concevoir l'algorithme d'optimisation distribuée asynchrone suivant.

Cet algorithme permet de s'affranchir d'un synchronisme à l'échelle du réseau pour le remplacer par un synchronisme local avec un ou quelques agents proches. Dans le cas typique où tous les sous-graphes consistent simplement en deux agents connectés, l'échange d'information et la synchronisation est très locale. Enfin, la convergence de cet algorithme est simplement une conséquence des deux théorèmes précédents.

Dans la Figure IV.2, nous trouvons l'erreur quadratique en fonction du nombre d'itération pour différents algorithmes d'optimisation distribuée. Le graphe des agents est représenté Figure 4.7 du manuscrit principal et leurs fonctions objectives sont quadratiques. Nous considérons une descente de gradient distribuée synchrone avec une mise à jour de Metropolis-Hastings, une descente de gradient distribuée asynchrone avec une mise à jour *Random Gossip*. Nous partitionons le graphe par paire d'agents comme dans la Figure 4.8 du manuscrit principal et considérons l'algorithme d'optimisation distribuée synchrone basé sur l'ADMM et notre algorithme asynchrone. Nous remarquons que notre algorithme offre de très bonne performances, presque comparables au cas synchrone, surtout si l'on considère le fait que chaque itération comporte deux argmin et un moyennage dans le cas asynchrone contre N argmin et 5 moyennages dans le cas synchrone.

Optimisation Distribuée Asynchrone basée sur l'ADMM

Soit ξ^{k+1} l'indice du sous-graphe qui s'active à l'instant k :

- Mise à jour primale des agents du sous-graphe :

$$\forall i \in V_{|\xi^{k+1}},\ x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho\sigma_i}{2} \left\| x - \frac{1}{\sigma_i} \left(\sum_{\ell \in \sigma_i} \bar{z}_{|\ell}^k - \frac{1}{\rho} \lambda_{i,|\ell}^{k+1} \right) \right\|^2 \right\}$$

- Le sous-graphe calcule sa moyenne locale :

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|V_{|\xi^{k+1}}|} \sum_{i \in V_{|\xi^{k+1}}} x_i^{k+1}$$

- Les agents du sous-graphe mettent à jour leur variable duale :

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^{k} + \rho(x_i^{k+1} - \bar{z}_{|\xi^{k+1}}^{k+1})$$

- Les autres agents ne font rien :

$$\forall i \notin V_{\xi^{k+1}},\ \ x_i^{k+1} = x_i^k \qquad \forall \ell \neq \xi^{k+1},\ \ z_{|\ell}^{k+1} = z_{|\ell}^k, \lambda_{|\ell}^{k+1} = \lambda_{|\ell}^k$$

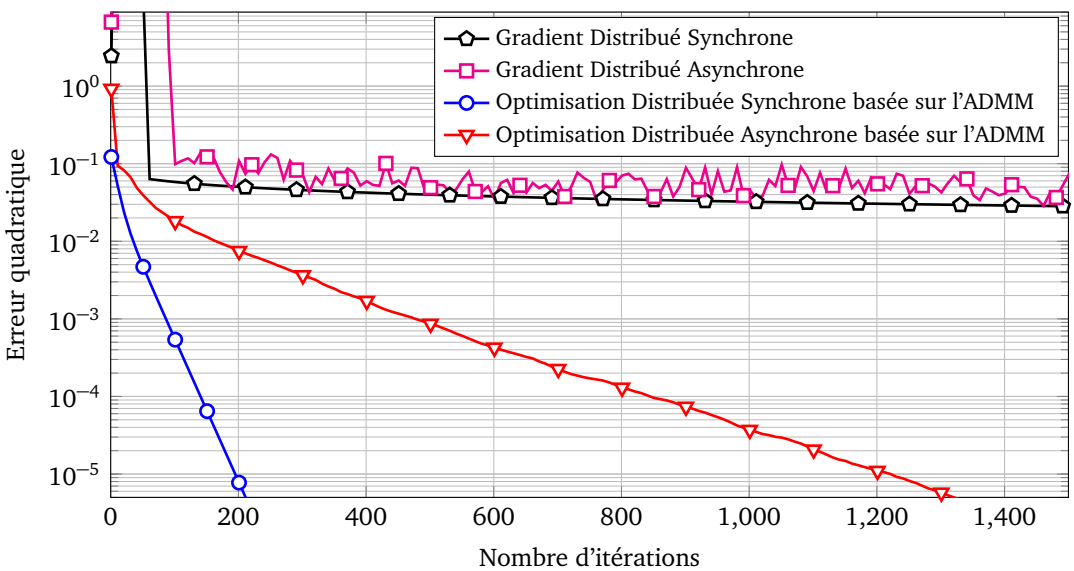


Figure IV.2 : Erreur quadratique en fonction du nombre d'itérations pour différents algorithmes d'optimisation distribuée.

Conclusion et Perspectives

Les travaux présentés dans cette thèse ont eu pour but d'analyser les algorithmes distribués asynchrones qui permettent à un réseau d'atteindre un consensus autour d'une valeur d'intérêt. Selon la nature de cette valeur (maximum, moyenne ou solution d'un problème d'optimisation), différents algorithmes ont été proposés et analysés.

Après avoir explicité notre modèle de réseau asynchrone basé sur la théorie des graphes, nous avons analysé comment propager le maximum des valeurs initiales. Deux algorithmes faisant intervenir des communications pair à pair pour l'un et *broadcast* pour l'autre ont été introduits et analysés en terme de temps de convergence. Nous avons notamment vu que l'utilisation de communications *broadcast* accélérait largement la propagation de l'information.

Ensuite, nous nous sommes concentrés sur le problème largement étudié du consensus autour de la moyenne des valeurs initiales. Dans ce contexte, l'utilisation de communications de type *broadcast* sans voie de retour ne permet pas de converger vers la valeur voulue. En revanche, nous avons montré que l'utilisation d'un modèle basé sur deux variables mises à jour conjointement, appelé *Sum-Weight*, permettait de résoudre ce problème. Nous avons prouvé des résultats généraux sur la convergence et la vitesse de convergence des algorithmes de moyennage basés sur ce modèle. Nous avons également conçu un algorithme de moyennage basé sur ce modèle et utilisant des communications de type *broadcast* sans voie de retour qui surpasse en performance les algorithmes de la littérature. Finalement, nous avons remarqué par simulation que les bornes proposées sur la vitesse de convergence étaient très fines, que ce soit pour notre algorithme ou pour les algorithmes standards de la littérature.

Enfin, nous avons étudié le problème de l'optimisation distribuée. À partir de l'algorithme ADMM dont les qualités de convergence et d'adaptabilité à notre problème ont été démontrées dans la littérature, nous avons utilisé le formalisme des opérateurs monotones pour proposer et prouver la convergence d'un algorithme d'optimisation distribuée asynchrone.

Perspectives. Notre analyse des problèmes du maximum et de la moyenne ont été importants pour notre compréhension de la diffusion d'information dans les réseaux ; toutefois, ces sujets ne sont pas prioritaires pour de futures recherches.

En revanche, le formalisme et l'algorithme asynchrone développés dans la dernière partie se sont montrés très prometteurs. L'étude des conditions et du taux pour la convergence linéaire (exponentielle) d'algorithmes d'optimisation distribuée basés sur l'ADMM est un de nos axes majeurs, il permettrait en outre d'obtenir un moyen de choisir le paramètre libre ρ de l'ADMM. Une autre piste d'étude est d'examiner la convergence de tels algorithmes lorsqu'une erreur est autorisée dans les étapes d'argmin, il y a alors un compromis entre la vitesse de convergence et l'erreur finale du réseau. Afin d'étudier les gains en termes de congestion du réseau apportés par l'asynchronisme, un implémentation pratique serait également très utile. Finalement, le formalisme des opérateurs monotones pourrait également nous permettre de concevoir des algorithmes asynchrones d'optimisation basés sur d'autres méthodes que l'ADMM comme le PPXA.

BIBLIOGRAPHY

[1] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[2] D. Spielman, "Spectral Graph Theory." Lecture Notes, 2009.

[3] R. Horn and C. Johnson, *Matrix analysis*. Cambridge university press, 2005.

[4] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[5] F. Chung, "Laplacians and the Cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[7] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast Gossip Algorithms for Consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[8] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer, 1999.

[9] S. Chatterjee and E. Seneta, "Towards consensus: some convergence theorems on repeated averaging," *Journal of Applied Probability*, vol. 14, pp. 89–97, 1977.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 1–12, 1987.

[11] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Analysis and optimization of randomized gossip algorithms," in *IEEE Conference on Decision and Control (CDC)*, pp. 5310–5315, 2004.

[12] P. Slater, E. Cockayne, and S. Hedetniemi, "Information dissemination in trees," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 692–701, 1981.

[13] B. Pittel, "On spreading a rumor," *SIAM Journal on Applied Mathematics*, vol. 47, no. 1, pp. 213–223, 1987.

[14] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, "Randomized broadcast in networks," *Random Structures and Algorithms*, vol. 1, no. 4, pp. 447–460, 1990.

[15] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *ACM-IEEE International Conference on Mobile Computing and Networking (SIGMOBILE)*, pp. 174–185, 1999.

[16] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 565–574, 2000.

[17] S. Kashyap, S. Deb, K. Naidu, R. Rastogi, and A. Srinivasan, "Efficient gossip-based aggregate computation," in *ACM Symposium on Principles of Database Systems (PODS)*, pp. 308–317, 2006.

[18] B. Doerr, T. Friedrich, and T. Sauerwald, "Quasirandom rumor spreading: Expanders, push vs. pull, and robustness," *Automata, Languages and Programming*, pp. 366–377, 2009.

[19] N. Fountoulakis, A. Huber, and K. Panagiotou, "Reliable Broadcasting in Random Networks and the Effect of Density," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–9, 2010.

[20] I. Chlamtac and S. Kutten, "On Broadcasting in Radio Networks–Problem Analysis and Protocol Design," *IEEE Transactions on Communications*, vol. 33, no. 12, pp. 1240–1246, 1985.

[21] O. Bar-Yehuda, R. Goldreich and A. Itai, "On the time-complexity of broadcast in radio networks: an exponential gap between determinism randomization," *Journal of Computer and System Sciences*, vol. 45, no. 1, pp. 104–126, 1992.

[22] E. Kushilevitz and Y. Mansour, "An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks," *SIAM Journal on Computing*, vol. 27, no. 3, pp. 702–712, 1998.

[23] L. Chrobak, M. Gasieniec and W. Rytter, "Fast broadcasting and gossiping in radio networks," *Journal of Algorithms*, vol. 43, no. 2, pp. 177–189, 2002.

[24] A. Czumaj and W. Rytter, "Broadcasting algorithms in radio networks with unknown topology," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 492–501, 2003.

[25] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.

[26] U. Feige, "A tight upper bound on the cover time for random walks on graphs," *Random Structures and Algorithms*, vol. 6, pp. 51–54, 1995.

[27] C. Cooper and A. Frieze, "The cover time of random geometric graphs," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 48–57, 2009.

[28] T. Sauerwald and A. Stauffer, "Rumor spreading and vertex expansion on regular graphs," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 462–475, 2011.

[29] D. Aldous and J. Fill, "Reversible Markov Chains and Random Walks on Graphs." Draft available online: http://www.stat.berkeley.edu/ aldous/RWG/book.html.

[30] C. Avin and G. Ercal, "On the cover time and mixing time of random geometric graphs," *Theoretical Computer Science*, vol. 380, no. 1-2, pp. 2–22, 2007.

[31] D. Gillman, "A Chernoff Bound for Random Walks on Expander Graphs," *SIAM Journal on Computing*, vol. 27, no. 4, pp. 1203–1220, 1998.

[32] A. Frieze and G. Grimmett, "The shortest-path problem for graphs with random arc-lengths," *Discrete Applied Mathematics*, vol. 10, no. 1, pp. 57–77, 1985.

[33] G. Giakkoupis and T. Sauerwald, "Rumor spreading and vertex expansion," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1623–1641, 2012.

[34] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995.

[35] M. Penrose, *Random geometric graphs*. Oxford University Press, USA, 2003.

[36] F. Bénézit, *Distributed Average Consensus for Wireless Sensor Networks*. PhD thesis, EPFL, 2009.

[37] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic Gossip: Efficient Averaging for Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, 2008.

[38] F. Benezit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Order-Optimal Consensus Through Randomized Path Averaging," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5150–5167, 2010.

[39] B. Nazer, A. G. Dimakis, and M. Gastpar, "Neighborhood gossip: Concurrent averaging through local interference," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3657–3660, 2009.

[40] D. Ustebay, B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, "Greedy Gossip With Eavesdropping," *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3765–3776, 2010.

[41] S. Wu and M. G. Rabbat, "Broadcast Gossip Algorithms for Consensus on Strongly Connected Digraphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 3959–3971, 2013.

[42] F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted Gossip: Distributed Averaging using non-doubly stochastic matrices," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1753–1757, 2010.

[43] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 482–491, 2003.

[44] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[45] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation*. Old Tappan, NJ (USA); Prentice Hall Inc., 1989.

[46] L. Xiao, S. Boyd, and S. Lall, "Distributed Average Consensus with Time-Varying Metropolis Weights." Unpublished, 2006.

[47] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should i choose for my sensor network?," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 986–994, 2008.

[48] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[49] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[50] A. Tahbaz-Salehi and A. Jadbabaie, "A Necessary and Sufficient Condition for Consensus Over Random Networks," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, pp. 791–795, 2008.

[51] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed Averaging in Sensor Networks Based on Broadcast Gossip Algorithms," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 808–817, 2011.

[52] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum Sensing for Cognitive Radio : State-of-the-Art and Recent Advances," *IEEE Signal Processing Magazine*, vol. 29, no. 3, pp. 101–116, 2012.

[53] P. Braca, S. Marano, V. Matta, and P. Willett, "Asymptotic Optimality of Running Consensus in Testing Binary Hypotheses," *IEEE Transactions on Signal Processing*, vol. 58, no. 2, pp. 814–825, 2010.

[54] F. S. Cattivelli and A. H. Sayed, "Distributed Detection Over Adaptive Networks Using Diffusion Adaptation," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 1917–1932, 2011.

[55] W. Zhang, Z. Wang, Y. Guo, H. Liu, Y. Chen, and J. Mitola, "Distributed Cooperative Spectrum Sensing Based on Weighted Average Consensus," in *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–6, 2011.

[56] H. L. Van Trees, *Detection, estimation, and modulation theory*. Wiley, 2004.

[57] P. Ciblat, P. Bianchi, and M. Ghogho, "Training Sequence Optimization for Joint Channel and Frequency Offset Estimation," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3424–3436, 2008.

[58] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed Clustering Using Wireless Sensor Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.

[59] J. Tsitsiklis, *Problems in decentralized decision making and computation*. PhD thesis, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984.

[60] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[61] A. Nedić, D. Bertsekas, and V. Borkar, "Distributed Asynchronous Incremental subgradient methods," in *Studies in Computational Mathematics*, vol. 8, pp. 381–407, Elsevier, 2001.

[62] A. Nedić and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[63] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[64] P. Bianchi and J. Jakubowicz, "Convergence of a Multi-Agent Projected Stochastic Gradient Algorithm for Non Convex Optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 391 – 405, 2013.

[65] D. Jakovetić, J. M. F. Moura, and X. Joao, "Distributed Nesterov-like gradient algorithms," in *IEEE Conference on Decision and Control (CDC)*, pp. 5459–5464, 2012.

[66] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[67] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A distributed Newton method for network optimization," in *IEEE Conference on Decision and Control (CDC)*, pp. 2736–2741, 2009.

[68] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[69] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in Ad Hoc WSNs With Noisy Links – Part I: Distributed Estimation of Deterministic Signals," *IEEE Transactions on Signal Processing*, vol. 56, pp. 350 –364, jan. 2008.

[70] E. Wei and A. Ozdaglar, "Distributed Alternating Direction Method of Multipliers," in *IEEE Conference on Decision and Control (CDC)*, pp. 5445–5450, 2012.

[71] P. Bianchi, G. Fort, and W. Hachem, "Performance of a Distributed Stochastic Approximation Algorithm," *IEEE Transactions on Information Theory*, vol. PP, no. 99, 2013.

[72] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[73] A. Nedić and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[74] G. Pólya and G. Szegö, *Problems and Theorems in Analysis I: Series Integral Calculus Theory of Functions*. Classics in mathematics, Springer, 1998.

[75] M. Duflo, *Random Iterative Models*. Springer-Verlag Berlin Heidelberg, 1997. Translated from original French edition "Méthodes Récursives Aléatoires" published by Masson, Paris, 1990.

[76] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications.," in *Optimization Methods in Statistics* (J. Rustagi, ed.), pp. 233–257, Academic Press, New York, 1971.

[77] G. Morral, P. Bianchi, G. Fort, and J. Jakubowicz, "Distributed stochastic approximation: The price of non-double stochasticity," in *Asilomar Conference on Signals, Systems and Computers*, pp. 1473–1477, 2012.

[78] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[79] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[80] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[81] J. Eckstein and D. P. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, pp. 293–318, 1992.

[82] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[83] G. J. Minty, "On the maximal domain of a "monotone" function.," *The Michigan Mathematical Journal*, vol. 8, no. 2, pp. 135–137, 1961.

[84] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales," *Fundamenta Mathematicae*, vol. 3, no. 1, pp. 133–181, 1922.

[85] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, Springer, 2011.

[86] J. Douglas and H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Transactions of the American Mathematical Society*, vol. 82, pp. 421–439, 1956.

[87] P. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[88] M. Schmidt, N. L. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization." arXiv preprint 1109.2415, 2011.

[89] P. Machart, S. Anthoine, and L. Baldassarre, "Optimal computational trade-off of inexact proximal methods." arXiv preprint 1210.5034, 2012.

ESTIMATION ET OPTIMISATION DISTRIBUÉE POUR LES RÉSEAUX ASYNCHRONES

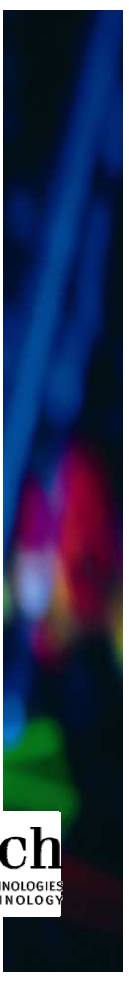
Franck IUTZELER

RESUME : Cette thèse s'intéresse au problème d'estimation et d'optimisation distribuée dans les réseaux asynchrones, c'est à dire en n'utilisant que des communication locales et asynchrones. A partir de multiples applications allant de l'apprentissage automatique aux réseaux de capteurs sans-fils, nous concevons et analysons théoriquement de nouveaux algorithmes résolvant trois problèmes de nature très différentes : la propagation de la plus grande des valeurs initiales, l'estimation de leur moyenne et enfin l'optimisation distribuée.

MOTS-CLEFS : Estimation Distribuée, Réseaux de Capteurs, Optimisation Distribuée

ABSTRACT : This thesis addresses the distributed estimation and optimization of a global value of interest over a network using only local and asynchronous (sometimes wireless) communications. Motivated by many different applications ranging from cloud computing to wireless sensor networks via machine learning, we design new algorithms and theoretically study three problems of very different nature : the propagation of the maximal initial value, the estimation of their average and finally distributed optimization.

KEY-WORDS : Distributed Estimation, Wireless Sensor Networks, Distributed Optimization



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY