TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Asynchronous Inference in Ambiently-Powered Wireless Sensor Networks

---

by Vasileios Papageorgiou

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DIPLOMA DEGREE OF

ELECTRICAL AND COMPUTER ENGINEERING

October, 2021

THESIS COMMITTEE
Professor Aggelos Bletsas, *Thesis Supervisor*
Professor George N. Karystinos
Associate Professor Michail G. Lagoudakis

Abstract

Wireless Sensor Networks (WSNs) are cost effective and ultra-low power networks that have recently become an integral part of many Internet-of-Things (IoT) applications. They consist of a certain number of *nodes* (or *terminals*), each of which is connected to a large number of sensors. Typically, the ambient information that the sensors are able to collect is wirelessly transferred to some kind of centralized processing unit, which usually involves *cloud* or *edge* technologies.

In this work, we consider a WSN that is *batteryless* and solely powered by the environment. Our goal is to utilize such a network removing the centralized processing unit, and, by carefully balancing the computation and communication cost of modern inference algorithms, allow it to make autonomous, in network decisions *itself*; all that, exploiting its *asynchronous* operation that stems from the fact that it is ambiently powered: at some point of time certain WSN *nodes* may fail to operate.

In particular, we consider a linear fixed point problem and mathematically formulate its *asynchronous* variant, aiming to capture the *asynchronous* operation of the WSN, according to which some parts of the calculated vector may not be updated at some iterations. We propose a *k-means* based clustering method of assigning different parts of a vector to different WSN *nodes*. Next, we describe two algorithms that are both expressed as linear fixed point problems: a) *Gaussian Belief Propagation* and b) *Average Consensus*, as well as their *asynchronous* variants introduced in this work. Analysis as well as numerical results of this work show that the *asynchronous* operation of a WSN can be a *key* in the convergence of Gaussian Belief Propagation; indeed, we show that different *asynchronous* schedulings vastly affect its convergence speed, and -in some cases- *asynchrony* can make a divergent instance (in synchronous operation) to converge. On the other hand, in the case of *Average Consensus*, we derive a statistical condition that, when satisfied, leads to *in expectation* convergence of the algorithm. Hence, it is possible to execute *Average Consensus* in an ambiently powered WSN; the caveat here is an increased delay, since independent repetitions of the algorithm are necessary for an accurate result.

Acknowledgements

A journey of 5 challenging yet exciting years comes to an end. During those wonderful years at the Technical University of Crete and the beautiful city of Chania, I was given the chance to grow both academically and as a person, something that is in a great extent owed to a special group of people.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Aggelos Bletsas, for our great collaboration, since he gave me the opportunity to work on a very interesting scientific field and -through his guidance- introduced me to research methodology. Secondly, I owe a lot to Professor Athanasios Liavas for not only introducing me to very intriguing and interesting scientific topics, but also for his invaluable mentorship. In addition, I would like to thank Professor Daphne Manoussaki for giving me the chance to gain valuable teaching assistance experience, and Professors Karystinos and Lagoudakis for evaluating my work. Finally, I would like to thank my colleagues, Thanos, Panagiotis and Giorgos, for our cooperation; it was amazing working with you!

Of course, nothing would be the same without my closest friends; our extended discussions as well as the fun times that we spent together have formed memories that I will *always* reminisce about.

Last, but definitely not least, nothing would be possible without my parents. Their support at every step of the way is what made me going, at every aspect of my life.

Thank you all!

-Vasileios Papageorgiou, October 2021

Contents

Chapter 1

Introduction

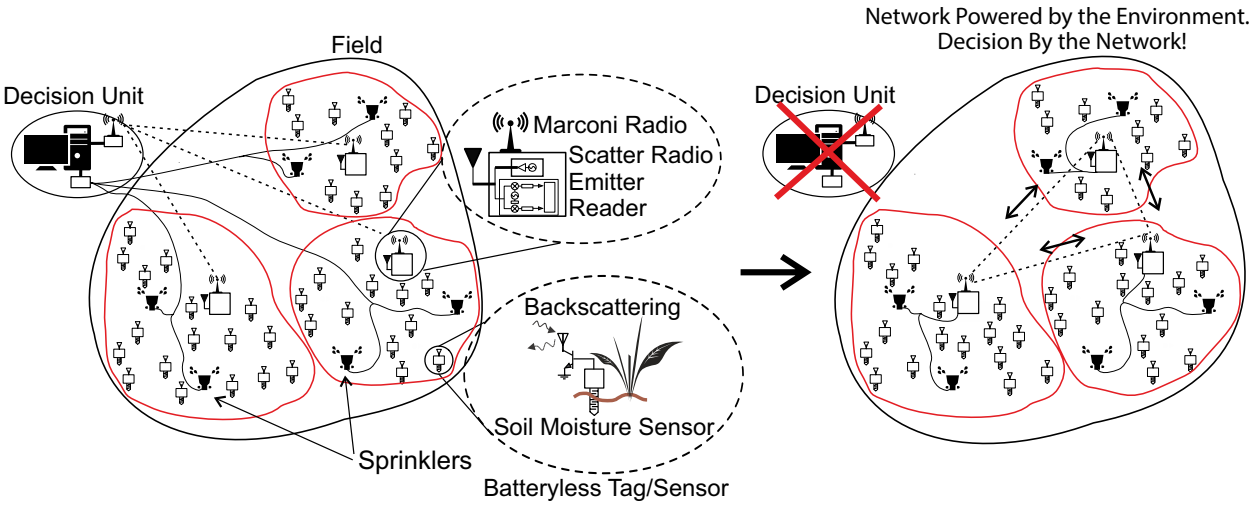


Figure 1.1: Vision: an ambiently powered, batteryless WSN operating on an agricultural field. The central decision unit/cloud service is removed and the network (that is solely powered by the environment) manages to process information utilizing message passing algorithms in order to decide *itself* where and when to irrigate!

The term *Wireless Sensor Networks* (WSN) refers to networks of sensors that collect ambient information and transfer it to a central processing unit. They consist of *nodes* (or *terminals*), each of which is connected to a large number of sensors. Such networks are particularly cost and power efficient. In particular, recent advances on backscatter radio sensor networks, have demonstrated feasibility of μWatt power and low-cost, joint sensing and wireless networking; [1, 2, 3, 4, 5, 6, 7, 8]; all is needed at the transmitter side is a radio frequency (RF) transistor, an antenna and a low-cost microcontroller unit.

The question that arises is the following: could a WSN that is solely powered by the environment be used *without* a central processing unit in order to make autonomous, in-network decisions? The answer to this question is positive, something that is in good part due to the network's *asynchronous*

operation [9].

In particular, advances on powerful message passing algorithms for inference and optimization [10, 11, 12] have offered novel tools on how different decision making and inference tasks can be accomplished through nodal communication in carefully constructed graphs. Our goal is to take advantage of the WSNs' and the algorithms' distributed nature and carefully balance the required computation and communication load across different WSN terminals, in order to exploit the power of the aforementioned algorithms and make in-network decisions [9]. For example, an ambiently powered WSN could collect different environmental parameters from various sensors and activate *itself* the appropriate sprinklers that are necessary to irrigate a field, without making use of any external processing unit or the cloud (Fig. 1.1).

However, one should not overlook the fact that the WSNs to which we refer are solely powered by the environment. Hence, there is the possibility that at some point, the ambient energy will not be sufficient. As a result, their operation is considered *asynchronous*, in the sense that they may fail to operate when they are required to. This very property is one of their main limitations when they are utilized in order to distributedly process information. However, in this work, we show that this is not the case; in particular, we show that *asynchrony* not only is *not* a limitation, but it can also become a key feature in the convergence rate of the message passing algorithms, or even in whether they will converge, in the first place! Therefore, we attempt to set the fundamental primitives for ambiently powered, batteryless WSNs that process the collected information themselves, namely Internet-of-Things (IoT)-That-Think.

Thesis Outline

In Chapter 2 we formulate the problem that we aim to solve, in Chapter 3 we provide two algorithmic frameworks that fit our system model, in Chapter 4 we state the major results regarding the convergence of the algorithms, in Chapter 5 we present the main numerical results and in Chapter 6 we state the main contributions of this work.

Notation

Vectors and matrices are denoted using bold lower-case and upper-case letters, respectively. x_i denotes the i-th entry of vector $\mathbf{x}$. In an iterative process $\mathbf{x}^{(l)} = f(\mathbf{x}^{(l\ \ 1)})$, $\mathbf{x}^{(l)}$ denotes the value of $\mathbf{x}$ at iteration (l). diag$f\mathbf{x}g$ denotes the diagonal matrix, whose diagonal entries are the elements of $\mathbf{x}$. $\mathbf{O}$ denotes the matrix whose all entries are zero. $\mathbf{x}_I$ denotes a subset of the elements of $\mathbf{x}$ according to the set of indices I, namely $\mathbf{x}_I \quad (x_1, x_2, \ldots, x_n)$. $\mathbf{A}_{ij}$ denotes the element of matrix $\mathbf{A}$ in its i-th row and j-th column. Given an undirected graph $G = (V, E)$, $\mathbf{nbr}(a)$ denotes the set of vertices that are adjacent to vertex a. $N(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C})$ denotes that a random variable follows the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$. $R(\mathbf{A})$ and $N(\mathbf{A})$ denote the rangespace and nullspace of matrix $\mathbf{A}$, respectively. $\dim V$ denotes the dimension of vector space V, namely the number of vectors in any of its bases. $\rho(\mathbf{A})$ denotes the spectral radius of matrix $\mathbf{A}$, namely $\rho(\mathbf{A}) = \max f j\lambda_1 j, j\lambda_2 j, \ldots, j\lambda_m j g$, where $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of $\mathbf{A}$.

Chapter 2

Asynchrony

In this chapter, we describe the system model and what we refer to as *asynchronous* operation. In particular, we describe the fixed point problem that we aim to solve and formulate a specific asynchronous variant of it. In addition, we describe how the operation of WSNs fits the aforementioned model and suggest a method of assigning different parts of the computation to different WSN terminals.

2.1 System Model

Let the real vectors $\mathbf{x}^{(0)}$, $\mathbf{b}$ 2 R^n and the real rectangular matrix $\mathbf{A}^{n \quad n}$. A vast majority of inference algorithms, including *Gaussian Belief Propagation* and *Average Consensus* that we will discuss in the sequel, can be formulated as the recursion

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l \quad 1)} + \mathbf{b}, \qquad l = 1, 2, \ldots. \tag{2.1}$$

In this work, our goal is to find the *xed point* of Eq. 2.1, namely the vector $\mathbf{x} \quad , \lim_{l \, ! \, + \, 1} \mathbf{x}^{(l)}$.

2.2 Asynchrony Formulation

Taking a closer look at Eq. 2.1, we can see that at iteration (l) the elements of $\mathbf{x}^{(l)}$ are calculated as

$$x_k^{(l)} = \sum_{j=1}^{n} a_{kj} x_j^{(l \quad 1)} + b_k, \qquad l = 1, 2, \ldots, \qquad k = 1, 2, \ldots, n, \tag{2.2}$$

where $x_k^{(l)}$ and b_k denote the k-th element of $\mathbf{x}^{(l)}$ and $\mathbf{b}$, respectively, and a_{ij} the element of i-th row and j-th column of $\mathbf{A}$. We can see that in order to compute the k-th element of $\mathbf{x}^{(l)}$, the knowledge of all $x_t^{(l \quad 1)}$ from the previous iteration $(l \quad 1)$ is required, where t 2 $f1, 2, \ldots, ng$ with respective

$a_{kt} \neq 0$. This requirement leads to *synchronous scheduling* in the sense that at all iterations, all the necessary values of the previous iteration are readily available when computing the elements of $\mathbf{x}^{(l)}$.

This strong requirement can be loosened. In particular, we assume that at iteration (l) some elements of $\mathbf{x}^{(l-1)}$ may not be available for the computation of the corresponding elements of the updated vector $\mathbf{x}^{(l)}$; in that case, we allow the latter not to be updated and keep the corresponding values of the previous iteration. In other words, it is possible for some elements $x_k^{(l)}$ to either get updated using Eq. 2.2 or directly keep the previous value $x_k^{(l-1)}$. This is what from now on we will refer to as *asynchronous scheduling*.

In order to formally formulate the above, we adopt the notation of seminal work in [13]. More specifically, at iteration (l), we introduce the functions $\psi_k^{(l)}$, $\forall k \in \{1,2\ldots,n\}$ which are defined as

$$\psi_k^{(l)} = \begin{cases} 1, & \text{if } x_k \text{ is updated at iteration } (l), \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Let $\boldsymbol{\psi}^{(l)}$ denote the binary vector that is created if we stack all $\psi_k^{(l)}$ at iteration (l) and $\boldsymbol{\Psi}^{(l)}$, $\text{diag}\{\boldsymbol{\psi}^{(l)}\}$ the corresponding diagonal matrix with $\boldsymbol{\psi}^{(l)}$ at its diagonal. As a result, the asynchronous framework that we previously discussed requires the recursion of Eq. 2.1 to become

$$\begin{aligned} \mathbf{x}^{(l)} &= \boldsymbol{\Psi}^{(l)} \left(\mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b} \right) + \left(\mathbf{I} - \boldsymbol{\Psi}^{(l)} \right) \mathbf{x}^{(l-1)} \\ &= \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \boldsymbol{\Psi}^{(l)} \right) \mathbf{x}^{(l-1)} + \boldsymbol{\Psi}^{(l)}\mathbf{b}. \end{aligned} \tag{2.4}$$

Notice that if $\boldsymbol{\Psi}^{(l)} = \mathbf{I}\ \forall l$, or equivalently $\psi_k^{(l)} = 1\ \forall l$ and $k \in \{1,2,\ldots,n\}$, the asynchronous update of Eq. 2.4 reduces to the synchronous one of Eq. 2.1.

In addition, we assume that the aforementioned functions $\psi_k^{(l)}$ are Bernoulli random variables. In [13], $\psi_k^{(l)}$ are assumed independent and identically distributed across both (l) and k with the same parameter p. In contrast, in this work we assume a more general framework where $\psi_k^{(l)}$ are independent across the different iterations (l) but possibly dependent and not identically distributed Bernoulli random variables across k with parameters p_k, $\forall k \in \{1,2,\ldots,n\}$. As a result, we can define the expected value of matrices $\boldsymbol{\Psi}^{(l)}$, $\mathbb{E}[\boldsymbol{\Psi}^{(l)}]$, $\mathbf{P} = \text{diag}\{\mathbf{p}\}$, where $\mathbf{p} = \mathbb{E}[\boldsymbol{\psi}^{(l)}]$, a quantity that as we will see plays a major role in the convergence of recursion Eq. 2.4.

2.3 Wireless Sensor Networks (WSN) for Distributed Computations

Consider an ambiently powered Wireless Sensor Network (WSN) with N physical terminals. Our goal is to utilize such a network in order to solve the fixed point problem that we previously

discussed. In particular, following Eq. 2.1, we assume that each terminal i is responsible for the calculation of a unique subset of the elements of $\mathbf{x}^{(l)}$, denoted by $\mathbf{x}_{I_i}^{(l)}$, where $\backslash_{i=1}^{N} I = ;$ and $[_{i=1}^{N} I_i = f1, 2, \ldots, ng$. In that context, if we take a closer look at Eq. 2.2, we shall point out that the calculation of an element of $\mathbf{x}^{(l)}$ may require access to elements that are allocated to different WSN terminals. Hence, communication between the aforementioned terminals is required.

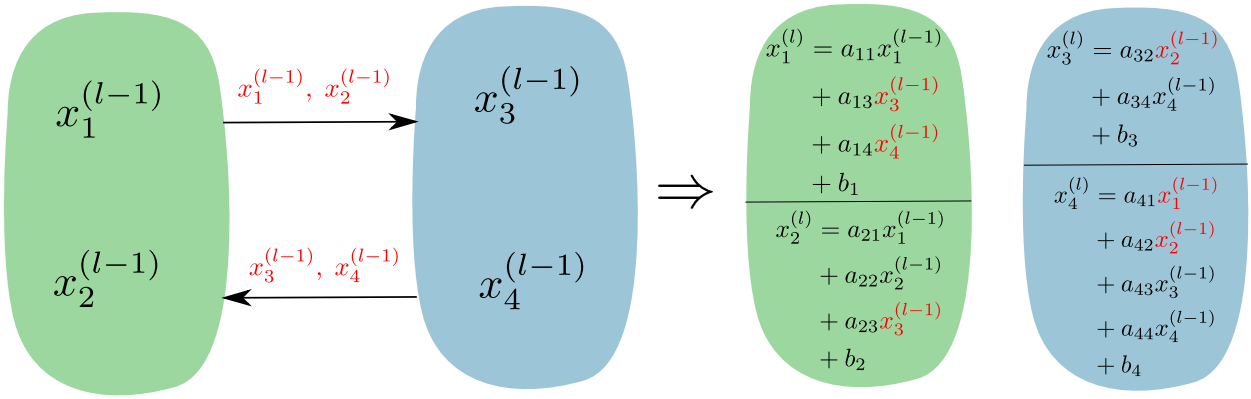


Figure 2.1: An update example for $\mathbf{x}\ 2\ \mathsf{R}^4$, $\mathbf{A} = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix}^{>}$ and a WSN with 2 terminals. We see that x_1 and x_2 are assigned to the left WSN terminal while x_3 and x_4 to the right one. Based on the non zero elements of $\mathbf{A}$, before any calculation is performed, the left terminal has to transmit its values to the right one and vice versa, something that is presented in the left part of the Figure; after the transmissions, on the right part of the Figure one can observe the update of each element of $\mathbf{x}$, as Eq. 2.2 requires.

In order to make this framework clearer, in Figure 2.1 we depict an update example for a WSN with 2 terminals and $\mathbf{x}\ 2\ \mathsf{R}^4$, with $\mathbf{A}$ and $\mathbf{b}$ of Eq. 2.1 given as

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

and

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

We can see that x_1 and x_2 are assigned at the left terminal while x_3 and x_4 are assigned at the right one. Based on the choice of $\mathbf{A}$, we observe that the calculation of the updated value of $x_1^{(l)}$ requires $x_3^{(l\ \ 1)}$ and $x_4^{(l\ \ 1)}$, two values that belong to a different WSN terminal; similarly, $x_2^{(l)}$ requires $x_3^{(l\ \ 1)}$,

$x_3^{(l)}$ requires $x_2^{(l\ \ 1)}$ and $x_4^{(l)}$ requires $x_1^{(l\ \ 1)}$ and $x_2^{(l\ \ 1)}$. As a result, those required values must firstly be transmitted from a WSN terminal to another, before any calculation is made; then, after the completion of all transmissions, every element of $\mathbf{x}$ is updated using Eq. 2.2.

In addition, this distributed setting also perfectly fits the asynchronous formulation that we presented in the previous section. More specifically, since the WSN operates utilizing energy that directly stems from the environment, the aforementioned procedure may be interrupted due to energy insufficiency. As a consequence, some elements of $\mathbf{x}$ may fail to get updated. This can be perfectly modeled with Eq. 2.4, with the statistics of the random matrices $\mathbf{\Psi}^{(l)}$ being directly determined by the statistics of energy sufficiency.

2.3.1 Vector Clustering

A natural question that arises is how to assign different parts of vector $\mathbf{x}$ to the different WSN terminals. Although there are many approaches to solve this problem, each satisfying different criteria, in this work we consider an approach based on *k-means* algorithm. *K-means* algorithm was initially introduced by Macqueen in 1967 [14], and is an algorithm that separates a given set of points in a metric space to k distinct clusters.

In particular, let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ be a collection of vectors in a metric space (A, d), with $\mathbf{x}_i \ 2 \ A, \ 8i = f1, 2, \ldots, ng$, where $d(\ ,\ )$ the associated metric with A. In addition, let $\mathbf{C} = f\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k g$ be a clustering partition of $\mathbf{X}$, where $\mathbf{C}_i \quad \mathbf{X}$, with $\mathbf{C}_i \ 6= \ ;, \ 8i = f1, 2, \ldots, kg$, $\setminus_{i=1}^{k} \mathbf{C}_i \ 6= \ ;$ and $\lceil_{i=1}^{k} \mathbf{C}_i = \mathbf{X}$, where the number of desired clusters $k \quad n$ is known. The goal of k-means is to find a clustering partition $\mathbf{C}$, so that the total distances within the same clusters are minimal. Formally, the objective is to solve the optimization problem

$$\begin{aligned} \arg\min_{\mathbf{C}} \quad & \sum_{i=1}^{k} \sum_{\mathbf{x} 2 \mathbf{C}_i} d(\mathbf{x}, \boldsymbol{\mu}_i) \\ \text{subject to} \quad & \mathbf{C}_i \ 6= \ ;, \qquad 8i = 1, 2, \ldots n, \\ & \setminus_{i=1}^{k} \mathbf{C}_i = \ ;, \\ & \lceil_{i=1}^{k} \mathbf{C}_i = \mathbf{X}, \end{aligned} \tag{2.5}$$

where $\boldsymbol{\mu}_i$ is the average of all points within $\mathbf{C}_i$, called *centroid*. In Algorithm 1 we describe the steps of k-means that solves Problem 2.5. We can see that at each iteration, each set $\mathbf{C}_i$ is assigned with the points of $\mathbf{X}$ that are "closest" to the average $\boldsymbol{\mu}_i$, with the latter being re-calculated; the procedure is repeated until convergence.

Now, let's focus on the problem that we described in the previous Sections. Assume that our WSN has k different terminals. Given $\mathbf{A}$ and $\mathbf{b}$ of the system model in Eq. 2.1, our goal is to divide vector $\mathbf{x}$ into k disjoint clusters. Afterwards, the elements of each cluster $\mathbf{C}_i$ will be assigned to WSN terminal i.

In particular, consider the update of each element x_i described by Eq. 2.2. As we can see, the

Algorithm 1: K-means algorithm.

Initialize $\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \ldots, \boldsymbol{\mu}_k^{(0)}$
for *t=0,1,2...* **do**
  for *i=1,2,...,k* **do**
    $\mathbf{C}_i^{(t+1)} = f\mathbf{x}_m : d(\mathbf{x}_m, \boldsymbol{\mu}_i^{(t)}) \quad d(\mathbf{x}_m, \boldsymbol{\mu}_j^{(t)}),\ 8j = f1, 2 \ldots, kgg$
    $\boldsymbol{\mu}^{(t+1)} = \frac{1}{j\mathbf{C}_i^{(t+1)}j} \sum_{\mathbf{x}2\mathbf{C}_i^{(t+1)}} \mathbf{x}$
  end
end

update of x_i requires the nonzero elements of the i-th row of matrix $\mathbf{A}$. In view of the above, let

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1^{>} \\ \vdots \\ \mathbf{s}_n^{>} \end{bmatrix} \ 2\ \mathsf{B}^{n \quad n} \tag{2.6}$$

denote the binary matrix whose entries are 1 when the corresponding element of $\mathbf{A}$ is nonzero and 0 otherwise, namely

$$\mathbf{S}_{ij} = \begin{cases} 1, & \mathbf{A}_{ij}\ 6\!\!\!-\ 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2.7}$$

where $\mathbf{s}_i^{>}$ denotes the i-th row of $\mathbf{S}$. For example, the matrix $\mathbf{S}$ that is constructed from $\mathbf{A}$ of the example in Fig. 2.1 is

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Hence, vectors $\mathbf{s}_i$ are indicative of which elements of $\mathbf{x}$ the update of x_i requires. As a result, it is natural to conclude that when $d(\mathbf{s}_i, \mathbf{s}_j)$ is relatively small, for some i, j and metric $d(\ ,\ )$, then the updates of the corresponding x_i and x_j will both require similar subsets of $\mathbf{x}$. Therefore, the communication requirements between the different WSN terminals become minimal.

Considering all the above, in order to find clusters $\mathbf{C}_i$ and as a result the appropriate assignment to the WSN terminals, we utilize k-means algorithm to the rows of $\mathbf{S}$, $(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)$, using the l_1-norm as the distance metric $d(\ ,\ )$, namely

$$d(\mathbf{x}, \mathbf{y})\ ,\ k\mathbf{x} \quad \mathbf{y}k_1, \qquad \mathbf{x}, \mathbf{y}\ 2\ \mathsf{B}^n. \tag{2.8}$$

Since the vectors $\mathbf{s}_i$ are binary, then l_1-norm is equivalent to the *Hamming distance*, namely the number of different bits of two vectors. For example, $k\mathbf{s}_i \quad \mathbf{s}_j k_1 = 3$ means that $\mathbf{s}_i$ and $\mathbf{s}_j$ differ in 3 positions; as a result, the updates of x_i and x_j require 3 different elements of $\mathbf{x}$, in addition -perhaps- to some other common entries of $\mathbf{x}$.

Chapter 3

Algorithms

In this Chapter we firstly introduce the basic types of *Probabilistic Graphical Models* and what *inference* is. We then describe two algorithms that lie into the system model that we presented in the previous Chapter. More specifically, we initially prove that a particular instance of *Gaussian Belief Propagation* results in a linear fixed point problem as described by Eq. 2.1. Afterwards, we consider its asynchronous variant along with the asynchronous version of the well known consensus algorithm, *Average Consensus*.

3.1 Gaussian Belief Propagation

3.1.1 Inference and Probabilistic Graphical Models

Consider the collection of n random variables

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \; 2 \; A_1 \quad A_2 \qquad A_n, \tag{3.1}$$

where each random variable x_i takes on a value in the set A_i, with $i = 1, 2, \ldots, n$. With the term *inference* we refer to answering specific queries about the distribution of $\mathbf{x}$ [15]. In particular, assume that we have access to the collection of observations

$$\mathbf{y} = (y_1, y_2, \ldots, y_m) \; 2 \; B_1 \quad B_2 \qquad B_m, \tag{3.2}$$

where - again - each y_i takes on a value on the set B_i, for $i = 1, 2, \ldots, m$. Given the aforementioned setup, the three main inference tasks are:

1. Calculating *posterior* beliefs:

$$\begin{aligned} p(\mathbf{x} j \mathbf{y}) &= \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \\ &= \frac{p(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{x} 2 A_1 \qquad A_n} p(\mathbf{x}, \mathbf{y})}. \end{aligned} \tag{3.3}$$

2. Calculate the *marginal* distribution for some subset of variables X_i:

$$p(X_i) = \sum_{\mathbf{x} \setminus X_i} p(\mathbf{x}). \tag{3.4}$$

3. Calculating the *maximum a posteriori (MAP)* estimator:

$$\begin{aligned} \hat{\mathbf{x}}(\mathbf{y}) &= \underset{\mathbf{x} \in A_1 \times \cdots \times A_n}{\operatorname{argmax}} \; p(\mathbf{x} | \mathbf{y}) \\ &= \underset{\mathbf{x} \in A_1 \times \cdots \times A_n}{\operatorname{argmax}} \; \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \\ &= \underset{\mathbf{x} \in A_1 \times \cdots \times A_n}{\operatorname{argmax}} \; p(\mathbf{x}, \mathbf{y}). \end{aligned} \tag{3.5}$$

Regarding the computational complexity of the aforementioned queries, assume for simplicity that both the spaces of $\mathbf{x}$ and $\mathbf{y}$ have the same dimensions, namely $n = m$, and that $|X_i| = 1$ in Eq. 3.4. Moreover, assume that A_i and B_i are countable (this assumption leads to discrete random variables) and $A_i = B_i = Z$ for some set Z (i.e. the set of integers $\mathbb{Z}$, or some finite subset of $\mathbb{Z}$), for all $i = 1, 2, \ldots, n$. As a result,

$$Z^n = A_1 \qquad A_n = B_1 \qquad B_n. \tag{3.6}$$

In that case, the calculation of the denominator of Eq. 3.3,

$$p(\mathbf{y}) = \sum_{(x_1, x_2, \ldots, x_n) \in Z^n} p(\mathbf{x}, \mathbf{y}), \tag{3.7}$$

requires $|Z|^{2n}$ operations, since $|Z|^n$ summations are required for each of the $|Z|^n$ possible values of $\mathbf{y}$. With alike arguments, we can see that the marginalization of Eq. 3.4 requires $|Z|^n$ operations. Similarly, if no structure is utilized, the optimization problem of the MAP estimator in Eq. 3.5 requires searching over the entire space Z^n, for each possible value of $\mathbf{y}$, resulting in $|Z|^{2n}$ calculations.

However, in a different scenario, suppose that $(x_1, x_2, \ldots, x_n)$ are independent. In that case, the calculation of Eq. 3.7 becomes

$$\begin{aligned} p(\mathbf{y}) &= \sum_{(x_1, x_2, \ldots, x_n) \in Z^n} p(\mathbf{x}, \mathbf{y}) \\ &= \sum_{(x_1, x_2, \ldots, x_n) \in Z^n} p(x_1, \ldots, x_n, \mathbf{y}) \\ &= \sum_{x_1 \in Z} p(x_1, \mathbf{y}) \sum_{x_2 \in Z} p(x_2, \mathbf{y}) \cdots \sum_{x_n \in Z} p(x_n, \mathbf{y}), \end{aligned} \tag{3.8}$$

a calculation that requires $n|Z|$ operation for each of the $|Z|^n$ different values of $\mathbf{y}$, resulting in the total $n|Z|^{n+1}$ computations. In a similar manner, the calculation of Eq. 3.4 is found to require

$(n \quad 1)jZj^2$ operations and the solution of the optimization problem of Eq. 3.5 becomes separable, namely

$$\begin{aligned}\hat{\mathbf{x}}(\mathbf{y}) &= \underset{\mathbf{x}2Z^n}{\operatorname{argmax}}\, p(\mathbf{x},\mathbf{y}) \\ &= \underset{x_i 2Z}{\operatorname{argmax}}\, p(x_i,\mathbf{y}), \qquad i = 1,2,\ldots,n,\end{aligned} \tag{3.9}$$

while its number of computations can be calculated to be $njZj^{n+1}$.

Hence, one can conclude that enforcing structure to a distribution can heavily benefit the inference's computational complexity. *Probabilistic graphical models (PGMs)* provide an elegant mechanism for exploiting such structures in complex probability distributions, allowing them to be constructed and utilized effectively [15]. In particular, they consist of graph-like representations, where the nodes and their connectivity encapsulate the probabilistic properties of the distribution of interest and dictate a particular factorization of its distribution function. The three main classes of PGMs are described below.

Directed Graphical Models

Directed Graphical Models (or *Bayesian Networks*) consist of a directed acyclic graph $G = (V,E)$, where V are its vertices, which represent the random variables of the distribution, and $E \quad V \quad V$ the directed edges between them. The connectivity of G encapsulates conditional independence properties of the distribution of interest. More specifically, since G is directed and acyclic, a topological order can be found. Given such a topological order, it holds that [16]

$$x_i \;?\; \mathbf{x}_{\mathrm{pred}(i)n\,\mathrm{pa}(i)} j \mathbf{x}_{\mathrm{pa}(i)}, \tag{3.10}$$

where $\mathrm{pa}(i)$ denotes the indices of the parents of node x_i and $\mathrm{pred}(i)$ its predecessors in the ordering. In other words, given its parents, x_i is independent of all x_j that come before it in the topological ordering, except -of course- for its parents. In order to find all conditional independences of the given graphical model, algorithms that are based on *d-separation* can be utilized[1].

All things considered, given the graph $G = (V,E)$ of the model, the distribution function that is to be represented is factorized as

$$p(\mathbf{x}) = \prod_{i=1}^{jVj} p(x_i j \mathbf{x}_{\mathrm{pa}(i)}). \tag{3.11}$$

To give a better insight of Eq. 3.11, consider the graph that is depicted in Fig. 3.1. Based on its topology, the joint distribution function that is represented by the aforementioned graph is

$$p(\mathbf{x}) = p(x_1)p(x_2 j x_1)p(x_3 j x_1)p(x_4 j x_2, x_3, x_5)p(x_5 j x_3). \tag{3.12}$$

[1] A thorough description of those algorithms is beyond the scope of this work. Details can be found in [16], [17].

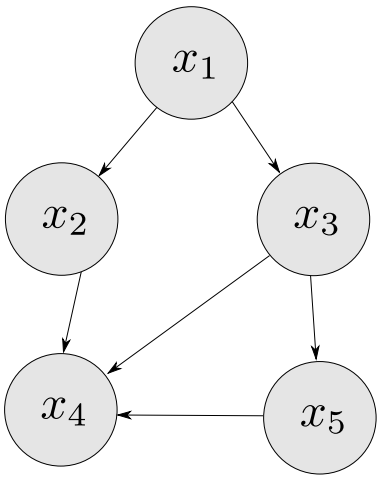


Figure 3.1: An example of a directed and acyclic graph that represents a distribution with 5 random variables.

It should also be highlighted that directed graphical models are considered *universal* in the sense that they can represent *any* distribution, since -based on the chain rule- every joint n-th dimensional distribution function can be written as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i j \setminus_{j=1}^{i} {}^{1} x_j), \tag{3.13}$$

a factorization that corresponds to a directed graphical model described by a fully connected directed and acyclic graph.

Undirected Graphical Models

Undirected Graphical Models (or *Markov Random Fields*) represent distributions using an undirected graph $G = (V, E)$, where V are its vertices, which represent the random variables, and $E \quad V \quad V$ the edges between them. Similarly to the case of Bayesian Networks, the connectivity of G describes conditional independence properties of the distribution of interest. More specifically, if $X, Y, Z \quad V$ denote the indices of three disjoint subsets of V, then

$$\mathbf{x}_X \ ? \ \mathbf{x}_Z j \mathbf{x}_Y \tag{3.14}$$

if and only if Y separates X and Z in V. To put it differently, Eq. 3.14 holds if and only if after the removal of all nodes in Y, there is no path connecting any nodes in X and Z.

Regarding the joint distribution that is described by G, we shall first define what *potential functions* are. In particular, a potential function $\psi_c(\mathbf{x}_c)$ is a non-negative function over a maximal clique $c \quad C$ of G, where C denotes the set of all maximal cliques of G. Thus, according to the *Hammersley-Cli ord* theorem [16], a positive distribution $p(\mathbf{x}) > 0$ satisfies the conditional

independence properties of an undirected graph G if and only if it can be factorized as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c2C} \psi_c(\mathbf{x}_c) \\ / \prod_{c2C} \psi_c(\mathbf{x}_c), \tag{3.15}$$

where Z is a normalization constant

$$Z = \sum_{\mathbf{x}} \prod_{c2C} \psi_c(\mathbf{x}_c). \tag{3.16}$$

It is crucial to highlight that one of the main drawbacks of undirected graphical models is the presence of Z, since its computational complexity is exponential in the size of the model [17].

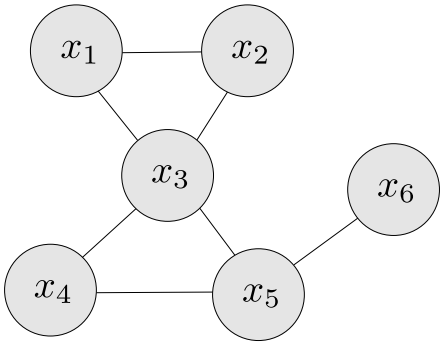


Figure 3.2: An example of an undirected graph that represents a distribution with 6 random variables.

In Fig. 3.2 we depict an example undirected graph that describes the conditional independences of a joint distribution with 6 random variables. The factorization that it enforces is

$$p(\mathbf{x}) \; / \; \psi_{f1,2,3g}(\mathbf{x}_{f1,2,3g}) \psi_{f3,4,5g}(\mathbf{x}_{f3,4,5g}) \psi_{f5,6g}(\mathbf{x}_{f5,6g}). \tag{3.17}$$

Factor Graphs

Factor graphs [11], [18] aim not only to unify directed and undirected graphical models' properties but also capture distributions' structures that the aforementioned graphical models fail to express. In particular, let $g : A \; ! \; R$ be an n-th dimensional real valued multivariate function of

$$\mathbf{x} \; , \; (x_1, x_2, \ldots, x_n) \; 2 \; A \; , \; A_1 \quad A_2 \qquad A_n. \tag{3.18}$$

Suppose that g can be factorized into a product of functions, each having a subset of $fx_1, x_2, \ldots, x_ng$ as arguments, namely

$$g(\mathbf{x}) = \prod_{i2I} f_i(\mathbf{x}_{J_i}), \tag{3.19}$$

where I and J_i are sets of indices, $\mathbf{x}_{J_i}$ $fx_1, x_2, \ldots, x_n g$ and f_i a real valued function that takes $\mathbf{x}_{J_i}$ as input. A factor graph $G = (V_x \ [\ V_f, E)$ is an undirected graph that expresses the factorization Eq. 3.19. More specifically, it contains *variable nodes* V_x - that represent the variables x_i- and *factor nodes* V_f -that represent the factors f_i of Eq. 3.19; the edges E $(V_x \ [\ V_f)$ $(V_x \ [\ V_f)$ only connect nodes *between* the sets V_x and V_f, and hence factor graphs are *bipartite*.

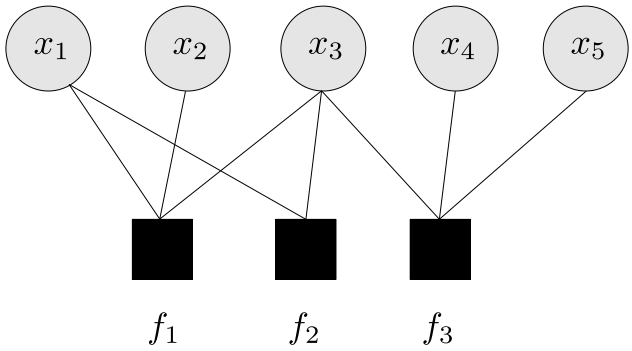


Figure 3.3: An example of a factor graph that corresponds to a function g of 5 variables factored as $g(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2, x_3) f_2(x_1, x_3) f_3(x_3, x_4, x_5)$.

In order to better comprehend what factor graphs are, we provide an example. Let g be a real valued function of 5 variables that can be factored as

$$g(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2, x_3) f_2(x_1, x_3) f_3(x_3, x_4, x_5). \tag{3.20}$$

In Fig. 3.3 we depict the factor graph associated with g. The variable nodes are represented with circles, while the factor nodes with black squares. As we can see, the constructed graph is bipartite, since there are no edges connecting two variable or two factor nodes.

Factor graphs are particularly useful describing probability distributions. In particular, if we assume that g refers to a distribution function, comparing Eq. 3.11 and Eq. 3.15 with Eq. 3.19 we observe that factor graphs can be seen as a generalization of both directed and undirected graphical models. On top of that, there is a variety of inference algorithms that have been developed taking advantage of the properties of factor graphs, algorithms that often involve *message passing* techniques between the graph's nodes. One of the most famous and widely used algorithms is the so called *sum-product* algorithm that we discuss in the next Section.

3.1.2 Belief Propagation Algorithm on Factor Graphs

Sum-product (or *belief propagation*) algorithm is a *message passing* algorithm that operates on undirected graphical models or factor graphs. It was firstly introduced by R. G. Gallager during the 1960s [19] and is still one of the most famous inference algorithms. Given a probability distribution function, sum-product computes -either exactly or approximately- various marginal distribution functions using message passing between the graph's nodes and following a few simple computational rules [11], [18].

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in A_1 \times A_2 \times \cdots \times A_n$ be a random vector and $p(\mathbf{x})$ its distribution function. Assume that a factor graph $G = (V_x \cup V_f, E)$ represents $p(\mathbf{x})$ according to the factorization

$$p(\mathbf{x}) = \prod_{i \in I} f_i(\mathbf{x}_{J_i}), \tag{3.21}$$

where I and J_i are sets of indices and $\mathbf{x}_{J_i} \in \{x_1, x_2, \ldots, x_n\}$, for all i. As we discussed in the previous Section, factor graphs are bipartite graphs; sum-product involves two different kinds of messages that are sent between graph's nodes: *variable-to-factor* and *factor-to-variable* messages. With $m_{x_i \to f_j}(x_i)$ we denote the message sent from random variable x_i to factor f_j and with $m_{f_j \to x_i}(x_i)$ the message sent from factor f_j to random variable x_i. According to sum-product, the computations of the aforementioned messages are

$$m_{x_i \to f_j}(x_i) = \prod_{g \in f_{\mathbf{nbr}(x_i)} \setminus f_j} m_{g \to x_i}(x_i) \tag{3.22}$$

$$m_{f_j \to x_i}(x_i) = \sum_{\mathbf{x}_{\mathbf{nbr}(f_j)} \setminus x_i} f_j(\mathbf{x}_{\mathbf{nbr}(f_j)}) \prod_{x_k \in \mathbf{x}_{\mathbf{nbr}(f_j)} \setminus x_i} m_{x_k \to f_j}(x_k), \tag{3.23}$$

where $\mathbf{nbr}(a)$ $(V_x \cup V_f)$ denotes the set of the neighboring nodes of a in the factor graph, in both the cases of random variable and factor nodes. Notice how both variable-to-factor and factor-to-variable messages are functions of the variable that they refer to. Then, the marginal distribution (or *belief*) of each random variable x_i of the model is calculated as

$$p(x_i) \propto \prod_{g \in \mathbf{nbr}(x_i)} m_{g \to x_i}(x_i). \tag{3.24}$$

As we mentioned earlier, sum-product may calculate marginal distributions either *exactly* or *approximately*. We initially consider the case of *tree* factor graphs. As the name suggests, tree factor graphs have the structure and properties of the general factor graphs that we discussed in the previous Section, but in addition, the associated graph is a *tree*. As a result, sum-product is executed *sequentially*; the messages that correspond to *leaf* nodes are initialized to units and the message passing operation starts from the aforementioned nodes to the internal ones. The algorithm naturally terminates after the messages from the internal nodes are finally passed to the leaf nodes and the beliefs are calculated; this operation results into the *exact* calculation of the marginal distributions.

The aforementioned procedure can be better comprehended with an example. Consider the tree factor graph that is depicted in Fig. 3.4 and can be found in [11]. Considering what we have discussed so far, and according to Eqs. 3.22, 3.23, the messages of sum-product are generated as follows:

1.

$$m_{f_1 \to x_1}(x_1) = \sum_{\mathbf{x}_{\mathbf{nbr}(f_1)} \setminus x_1 = ;} f_1(x_1) = f_1(x_1)$$

$$m_{f_2 \to x_2}(x_2) = \sum_{\mathbf{x}_{\mathbf{nbr}(f_2)} \setminus x_2 = ;} f_2(x_2) = f_2(x_2)$$

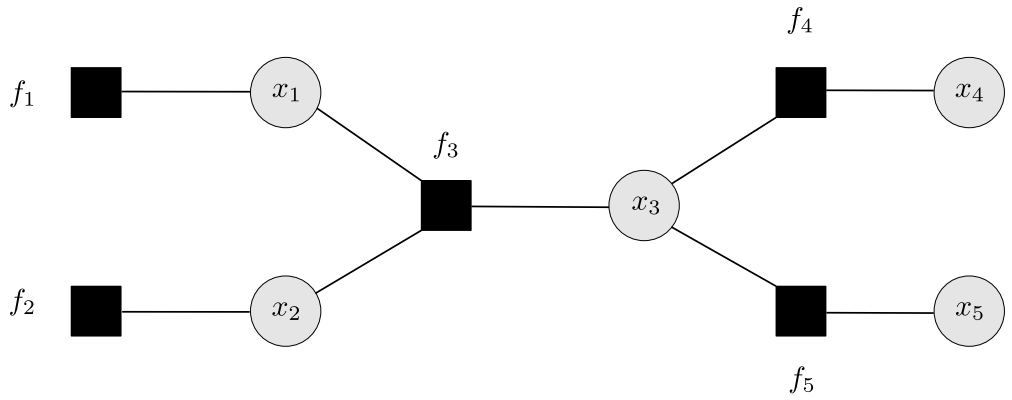


Figure 3.4: An example tree factor graph consisting of 5 random variable and 5 factor nodes [11].

$$m_{x_4 \,!\, f_4}(x_4) = 1$$
$$m_{x_5 \,!\, f_5}(x_5) = 1$$

2.

$$m_{x_1 \,!\, f_3}(x_1) = m_{f_1 \,!\, x_1}(x_1)$$
$$m_{x_2 \,!\, f_3}(x_2) = m_{f_2 \,!\, x_2}(x_2)$$
$$m_{f_4 \,!\, x_3}(x_3) = \sum_{x_4} m_{x_4 \,!\, f_4}(x_4) f_4(x_3, x_4)$$
$$m_{f_5 \,!\, x_3}(x_3) = \sum_{x_5} m_{x_5 \,!\, f_5}(x_5) f_5(x_3, x_5)$$

3.

$$m_{f_3 \,!\, x_3}(x_3) = \sum_{x_1, x_2} m_{x_1 \,!\, f_3}(x_1) m_{x_2 \,!\, f_3}(x_2) f_3(x_1, x_2, x_3)$$
$$m_{x_3 \,!\, f_3}(x_3) = m_{f_4 \,!\, x_3}(x_3) m_{f_5 \,!\, x_3}(x_3)$$

4.

$$m_{f_3 \,!\, x_1}(x_1) = \sum_{x_2, x_3} m_{x_3 \,!\, f_3}(x_3) m_{x_2 \,!\, f_3}(x_2) f_3(x_1, x_2, x_3)$$
$$m_{f_3 \,!\, x_2}(x_2) = \sum_{x_1, x_3} m_{x_3 \,!\, f_3}(x_3) m_{x_1 \,!\, f_3}(x_2) f_3(x_1, x_2, x_3)$$
$$m_{x_3 \,!\, f_4}(x_3) = m_{f_3 \,!\, x_3}(x_3) m_{f_5 \,!\, x_3}(x_3)$$
$$m_{x_3 \,!\, f_5}(x_3) = m_{f_3 \,!\, x_3}(x_3) m_{f_4 \,!\, x_3}(x_3)$$

5.

$$m_{x_1 \,!\, f_1}(x_1) = m_{f_3 \,!\, x_1}(x_1)$$
$$m_{x_2 \,!\, f_2}(x_2) = m_{f_3 \,!\, x_2}(x_2)$$

$$m_{f_4 \to x_4}(x_4) = \sum_{x_3} m_{x_3 \to f_4}(x_3) f_4(x_3, x_4)$$
$$m_{f_5 \to x_5}(x_5) = \sum_{x_3} m_{x_3 \to f_5}(x_3) f_5(x_3, x_5)$$

6.

$$p(x_1) = m_{f_1 \to x_1}(x_1) m_{f_3 \to x_1}(x_1)$$
$$p(x_2) = m_{f_2 \to x_2}(x_2) m_{f_3 \to x_2}(x_2)$$
$$p(x_3) = m_{f_3 \to x_3}(x_3) m_{f_4 \to x_3}(x_3) m_{f_5 \to x_3}(x_3)$$
$$p(x_4) = m_{f_4 \to x_4}(x_4)$$
$$p(x_5) = m_{f_5 \to x_5}(x_5).$$

Conversely, when the factor graph of interest has not a tree structure but contains loops, this sequential order of computations does not exist, since there is not the concept of *leaf* nodes. Therefore, sum-product takes a slightly different shape, called *loopy belief propagation*. In particular, loopy belief propagation contains the same message calculations as described in Eqs. 3.22 and 3.23. However, since there is no natural termination step as in the previous case, the aforementioned calculations are performed iteratively -after initialization- for a certain number of *iterations*, namely

$$m^{(t+1)}_{x_i \to f_j}(x_i) = \prod_{g \in f_{\mathbf{nbr}(x_i)} \setminus f_j} m^{(t)}_{g \to x_i}(x_i) \tag{3.25}$$
$$m^{(t+1)}_{f_j \to x_i}(x_i) = \sum_{\mathbf{x}_{\mathbf{nbr}(f_j)} \setminus x_i} f_j(\mathbf{x}_{\mathbf{nbr}(f_j)}) \prod_{x_k \in \mathbf{x}_{\mathbf{nbr}(f_j)} \setminus x_i} m^{(t)}_{x_k \to f_j}(x_k), \tag{3.26}$$

for $t = 1, 2, \ldots$. After termination, the marginal distributions are calculated using Eq. 3.24. In contrast to the case of tree factor graphs, loopy belief propagation may fail to converge. On top of that, the calculation of the beliefs becomes *approximate*.

3.1.3 Gaussian Belief Propagation under High-Order Factorization and Asynchronous Scheduling

Consider a Gaussian vector $\mathbf{x} \in \mathbb{R}^n$; its probability density function can be written as

$$p(\mathbf{x}) \propto \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Lambda}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \tag{3.27}$$

where $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ its expected value and $\boldsymbol{\Lambda} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top]$ its covariance matrix. A particularly useful representation of Gaussian distributions is the so called *information form*, where we can write

$$p(\mathbf{x}) \propto \exp\left\{ -\frac{1}{2}\mathbf{x}^{\top}\mathbf{J}\mathbf{x} + \mathbf{h}^{\top}\mathbf{x} \right\} \tag{3.28}$$

where $\mathbf{J} = \mathbf{\Lambda}^{-1} - \mathbf{O}$ its information (or precision) matrix and $\mathbf{h} = \mathbf{J}\boldsymbol{\mu}$ its potential vector. It is obvious that $p(\mathbf{x})$ can also be written as

$$p(\mathbf{x}) \propto \prod_{i=1}^{n} f_i(x_i) \prod_{j=1}^{m} g_j(\mathcal{X}_j), \tag{3.29}$$

where $f_i(x_i)$ are functions of x_i, $\forall i \in \{1, 2, \ldots, n\}$, and $g_j(\mathcal{X}_j)$ functions of the sets $\mathcal{X}_j \subseteq \{x_1, x_2, \ldots, x_n\}$, $\forall j \in \{1, 2, \ldots, m\}$ (we assume that there exist m such sets). If any of $\mathcal{X}_j$ contains more than two elements of $\mathbf{x}$, then we refer to Eq. 3.29 as a *high-order factorization* of the joint Gaussian PDF.

Following [13], in this work we consider a particular factorization of the information matrix and potential vector of Eq. 3.29. More specifically, we consider a high-order factorization with $\mathbf{J} = \mathbf{\Lambda} + \mathbf{\Xi}^{\top}\mathbf{\Sigma}\mathbf{\Xi}$ and $\mathbf{h} = \mathbf{\Lambda}\boldsymbol{\xi} + \mathbf{\Xi}^{\top}\mathbf{\Sigma}\mathbf{u}$, where $\mathbf{\Lambda} \succeq \text{diag}\{\eta_1, \eta_2, \ldots, \eta_n\}$, $\mathbf{\Sigma} \succeq \text{diag}\{\zeta_1, \zeta_2, \ldots, \zeta_m\}$ with $\eta_i \geq 0$, $\forall i \in \{1, 2, \ldots, n\}$, $\zeta_j > 0$, $\forall j \in \{1, 2, \ldots, m\}$, $\mathbf{\Xi} \in \mathbb{R}^{m \times n}$, $\boldsymbol{\xi} \in \mathbb{R}^{n}$ and $\mathbf{u} \in \mathbb{R}^{m}$. Under this factorization, the Gaussian PDF Eq. 3.28 takes the form

$$\begin{aligned} p(\mathbf{x}) &\propto \exp\left\{ -\frac{1}{2}\mathbf{x}^{\top}\left(\mathbf{\Lambda} + \mathbf{\Xi}^{\top}\mathbf{\Sigma}\mathbf{\Xi}\right)\mathbf{x} + \left(\mathbf{\Lambda}\boldsymbol{\xi} + \mathbf{\Xi}^{\top}\mathbf{\Sigma}\mathbf{u}\right)^{\top}\mathbf{x} \right\} \\ &\propto \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\xi})^{\top}\mathbf{\Lambda}(\mathbf{x} - \boldsymbol{\xi}) \right\} \exp\left\{ -\frac{1}{2}(\mathbf{\Xi}\mathbf{x} - \mathbf{u})^{\top}\mathbf{\Sigma}(\mathbf{\Xi}\mathbf{x} - \mathbf{u}) \right\} \\ &\propto \prod_{i=1}^{n} \exp\left\{ -\frac{1}{2}\eta_i (x_i - \xi_i)^2 \right\} \prod_{j=1}^{m} \exp\left\{ -\frac{1}{2}\zeta_i (\mathbf{\Xi}_{j,:}\mathbf{x} - u_j)^2 \right\}, \end{aligned} \tag{3.30}$$

where $\mathbf{\Xi}_{j,:}$ denotes the j-th row of matrix $\mathbf{\Xi}$. Comparing Eq. 3.30 with Eq. 3.29, one can observe that

$$f_i(x_i) = \exp\left\{ -\frac{1}{2}\eta_i (x_i - \xi_i)^2 \right\} \tag{3.31}$$

$$g_j(\mathcal{X}_j) = \exp\left\{ -\frac{1}{2}\zeta_i \left(\sum_{k \in \mathcal{V}_j} \Xi_{j,k} - u_j \right)^2 \right\} \tag{3.32}$$

where $\mathcal{X}_j = \{x_k | \Xi_{j,k} \neq 0\}$ and $\mathcal{V}_j = \{k | x_k \in \mathcal{X}_j\}$. As we discussed in the previous Sections, if we take a closer look at Eqs. 3.30, 3.31 and 3.32 we can see that a factor graph can be constructed, with f_i and g_j as factor nodes and x_i as random variable nodes, $\forall j \in \{1, 2, \ldots, m\}$ and $\forall i \in \{1, 2, \ldots, n\}$, respectively; as we examined, since factor graphs are bipartite graphs, sum-product algorithm - which in the case of Gaussian distributions is called *Gaussian Belief Propagation* - involves the iterative update of two types of nodal messages: factor-to-variable and variable-to-factor messages.

With $m^{(l)}_{g_j \to x_i}(x_i)$ we denote the message sent from factor g_j to random variable x_i and $m^{(l)}_{x_i \to g_j}(x_i)$ the one sent from random variable x_i to factor g_j, both at iteration (l). According to sum-product, the expressions of the aforementioned messages are

$$m^{(l)}_{g_j \to x_i}(x_i) \propto \int_{\mathbf{x}_j \setminus x_i} g_j(\mathbf{x}_j) \prod_{k \in \mathcal{V}_j \setminus i} m^{(l-1)}_{x_k \to g_j}(x_k)\, d\mathbf{x}_j \setminus x_i \tag{3.33}$$

$$m^{(l)}_{x_i \to g_j}(x_i) \propto f_i(x_i) \prod_{k \in \mathcal{G}_i \setminus j} m^{(l)}_{g_k \to x_i}(x_i), \tag{3.34}$$

where $\mathcal{G}_i$ denotes the set of indices of factors g_j, connected directly to random variable x_i in the factor graph. Using Eq. 3.34 into Eq. 3.33, the expression of factor-to-variable messages becomes

$$m^{(l)}_{g_j \to x_i}(x_i) \propto \int_{\mathbf{x}_j \setminus x_i} g_j(\mathbf{x}_j) \prod_{k \in \mathcal{V}_j \setminus i} \left(f_k(x_k) \prod_{k' \in \mathcal{G}_k \setminus j} m^{(l)}_{g_{k'} \to x_i}(x_k) \right) d\mathbf{x}_j \setminus x_i. \tag{3.35}$$

Notice how the expression Eq. 3.35 is only parametrized by factor-to-variable messages. Without loss of generality, we assume that the factor-to-variable messages $m^{(l-1)}_{g_{k'} \to x_i}(x_k)$ are of Gaussian form with $m^{(l-1)}_{g_{k'} \to x_i}(x_k) \propto \mathcal{N}\left(x_k; \mu^{(l-1)}_{g_{k'} \to x_i}, \frac{1}{v^{(l-1)}_{g_{k'} \to x_i}}\right)$, where $\mu^{(l-1)}_{g_{k'} \to x_i}$ and $v^{(l-1)}_{g_{k'} \to x_i}$ their mean and precision, respectively. Thus, substituting the expressions of $f_i(x_i)$ and $g_j(\mathbf{x}_j)$ presented in Eq. 3.31 and 3.32 and the Gaussian form of $m^{(l-1)}_{g_{k'} \to x_i}(x_k)$ in Eq. 3.35 we get that the analytical expression of factor-to-variable nodes

$$\begin{aligned} m^{(l)}_{g_j \to x_i}(x_i) \propto \int_{\mathbf{x}_j \setminus x_i} & \exp\left\{ -\frac{1}{2}\zeta_j \left(\sum_{k \in \mathcal{V}_j} \Xi_{jk} x_k - u_j \right)^2 \right\} \\ & \prod_{k \in \mathcal{V}_j \setminus i} \exp\left\{ -\frac{1}{2}\left(\eta_k + \sum_{k' \in \mathcal{G}_k \setminus k} v^{(l-1)}_{g_{k'} \to x_k} \right) x_k^2 + \left(\eta_k \xi_k + \sum_{k' \in \mathcal{G}_k \setminus j} v^{(l-1)}_{g_{k'} \to x_k} \mu^{(l-1)}_{g_{k'} \to x_k} \right) x_k \right\} d\mathbf{x}_j \setminus x_i. \end{aligned} \tag{3.36}$$

It can be proven [13] that the messages $m^{(l)}_{g_j \to x_i}(x_i)$ are maintained Gaussian at iteration (l), with their mean and precision given by

$$\mu^{(l)}_{g_j \to x_i} = \begin{cases} \Xi_{ji}^{-1} u_j - \sum_{k \in \mathcal{V}_j \setminus i} \dfrac{\Xi_{ji}^{-1} \Xi_{jk} \left(\eta_k \xi_k + \sum_{k' \in \mathcal{G}_k \setminus j} v^{(l-1)}_{g_{k'} \to x_k} \mu^{(l-1)}_{g_{k'} \to x_k} \right)}{\eta_k + \sum_{k' \in \mathcal{G}_k \setminus k} v^{(l-1)}_{g_{k'} \to x_k}}, \\ \qquad \text{if } \eta_k + \sum_{k' \in \mathcal{G}_k \setminus k} v^{(l-1)}_{g_{k'} \to x_k} > 0, \forall k \in \mathcal{V}_k \setminus i \\ 0, \qquad \text{otherwise} \end{cases} \tag{3.37}$$

$$v_{g_j \to x_i}^{(l)} = \frac{\Xi_{ji}^2}{\zeta_j^{-1} + \sum_{k \in V_j \setminus i} \Xi_{jk}^2 \left(\eta_k + \sum_{k' \in G_k \setminus k} v_{g_{k'} \to x_k}^{(l-1)} \right)^{-1}}, \tag{3.38}$$

respectively.

To continue, we can use the updated messages $m_{g_j \to x_i}^{(l)}$ and $f_i(x_i)$ in order to calculate the belief propagation belief $b^{(l)}(x_i)$ of x_i at iteration (l) as

$$b^{(l)}(x_i) \propto f_i(x_i) \prod_{k \in G_i} m_{g_k \to x_i}^{(l)}(x_i). \tag{3.39}$$

Again, we can substitute the expressions of $f_i(x_i)$ and $m_{g_k \to x_i}^{(l)}(x_i)$ into Eq. 3.39 and get

$$b^{(l)}(x_i) \propto \exp\left\{ -\frac{1}{2} \left(\eta_i + \sum_{k \in G_i} v_{g_k \to x_i}^{(l)} \right) x_i^2 + \left(\eta_i \xi_i + \sum_{k \in G_i} v_{g_k \to x_i}^{(l)} \mu_{g_k \to x_i}^{(l)} \right) x_i \right\}. \tag{3.40}$$

It is proven in [13] that $b^{(l)}(x_i)$ are valid Gaussians with $b^{(l)}(x_i) \sim \mathcal{N}\left(x_i; \epsilon_i^{(l)}, \sigma_i^{(l)}\right)$, where the mean and variance are given by

$$\epsilon_i^{(l)} = \frac{\eta_i \xi_i + \sum_{k \in G_i} v_{g_k \to x_i}^{(l)} \mu_{g_k \to x_i}^{(l)}}{\eta_i + \sum_{k \in G_i} v_{g_k \to x_i}^{(l)}} \tag{3.41}$$

$$\sigma_i^{(l)} = \frac{1}{\eta_i + \sum_{k \in G_i} v_{g_k \to x_i}^{(l)}}, \tag{3.42}$$

respectively.

With $\mathbf{v}^{(l)}$ and $\boldsymbol{\mu}^{(l)}$ we denote the vectors constructed if we stack all $v_{g_j \to x_i}^{(l)}$ and $\mu_{g_j \to x_i}^{(l)}$, $\forall (i,j) \in E$, $(i,j) \neq \{1,2,\ldots,n\}, j \in G_i\}$, respectively, with the order of (i,j) ascending first on i then on j. Additionally, we define Θ , $\left\{ \theta_{g_j \to x_i} \quad \Xi_{ji}^2 \zeta_j, \forall (i,j) \in E \right\}$, where $\boldsymbol{\theta}$ is a vector containing the elements $\theta_{g_j \to x_i}$, $\forall (i,j) \in E$ with the same order as in $\mathbf{v}^{(l)}$ and $\boldsymbol{\mu}^{(l)}$. As shown in [13], if $\mathbf{v}^{(0)}$ is chosen in a way that $\mathbf{v}^{(0)} \in \Theta$, then $\mathbf{v}^{(l)}$ converges to a unique non-negative fixed point $\mathbf{v}^*$, with either a *synchronous* or *asynchronous scheduling*, as described in Chapter 2.

Once the precision $\mathbf{v}^{(l)}$ converges, the update of belief propagation message means follows the expression Eq. 3.37, with the values $v_{g_{k'} \to x_k}^{(l-1)}$ being replaced by the converged values $v_{g_{k'} \to x_k}^*$. Taking a closer look at Eq. 3.37, we can see that the update of vector $\boldsymbol{\mu}^{(l)}$ is a linear function. In particular, it holds that

$$\boldsymbol{\mu}^{(l)} = \mathbf{A}\boldsymbol{\mu}^{(l-1)} + \mathbf{c}, \qquad l = 1, 2, \ldots, \tag{3.43}$$

where $\mathbf{A} \in \mathbb{R}^{|E| \times |E|}$ is a matrix such that $\mathbf{A}\boldsymbol{\mu}^{(l-1)}$ is a column vector containing elements $\alpha_{i,j}, \forall (i,j) \in E$ arranged in the same manner as in $\mathbf{v}^{(l)}$ and $\boldsymbol{\mu}^{(l)}$, with

$$\alpha_{i,j} = \begin{cases} \sum\limits_{k \in V_j \setminus i} \dfrac{\Xi_{ji}^{-1}\Xi_{jk}\left(\eta_k \xi_k + \sum_{k' \in G_k \setminus j} v_{g_{k'} \to x_k} \mu_{g_{k'} \to x_k}^{(l-1)}\right)}{\eta_k + \sum_{k' \in G_k \setminus k} v_{g_{k'} \to x_k}}, & \text{if } \eta_k + \sum\limits_{k' \in G_k \setminus k} v_{g_{k'} \to x_k} > 0, \forall k \in V_k \setminus i \\ 0, & \text{otherwise} \end{cases} \tag{3.44}$$

and

$$c_{i,j} = \begin{cases} \Xi_{ji}^{-1} u_j \sum\limits_{k \in V_j \setminus i} \dfrac{\Xi_{ji}^{-1}\Xi_{jk}\eta_k \xi_k}{\eta_k + \sum_{k' \in G_k \setminus k} v_{g_{k'} \to x_k}}, & \text{if } \eta_k + \sum\limits_{k' \in G_k \setminus k} v_{g_{k'} \to x_k} > 0, \forall k \in V_k \setminus i \\ 0, & \text{otherwise.} \end{cases} \tag{3.45}$$

As we can see, the belief propagation message mean update in Eq. 3.43 is an instance of the system model that we discussed in Chapter 2. Hence, as it was initially introduced in [13] and analyzed in the aforementioned Chapter, we can solve the fixed point problem of Eq. 3.43 in an *asynchronous* manner utilizing Eq. 2.4. This will lead to the *asynchronous* belief propagation message mean update

$$\boldsymbol{\mu}^{(l)} = \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right)\boldsymbol{\mu}^{(l-1)} + \boldsymbol{\Psi}^{(l)}\mathbf{c}. \tag{3.46}$$

3.1.4 Solving a linear system of equations

Suppose that our goal is to solve the linear system of equations

$$\mathbf{M}\mathbf{x} = \mathbf{s}, \tag{3.47}$$

where $\mathbf{M} \in \mathbb{R}^{m \times n}$ with $m \geq n$ a full rank matrix and $\mathbf{s} \in \mathbb{R}^m$ a real vector. As we know, the least-squares solution of Eq. 3.47 yields the vector

$$\mathbf{x} = \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{s}. \tag{3.48}$$

The aforementioned system can be solved using Gaussian Belief Propagation. More specifically, if we set $\boldsymbol{\Lambda} = \mathbf{O}$, $\boldsymbol{\Xi} = \mathbf{M}$, $\boldsymbol{\Sigma} = \mathbf{I}$, $\boldsymbol{\xi} = \mathbf{0}$ and $\mathbf{u} = \mathbf{s}$ in Eq. 3.30 then Gaussian Belief Propagation can be utilized for the following distribution

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \left(\mathbf{M}^{\top}\mathbf{M}\right)^{-1}\mathbf{M}^{\top}\mathbf{s}, \left(\mathbf{M}^{\top}\mathbf{M}\right)^{-1}\right) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^{\top}\mathbf{M}^{\top}\mathbf{M}\mathbf{x} + \mathbf{s}^{\top}\mathbf{M}\mathbf{x}\right\}, \tag{3.49}$$

whose inference of expected value will yield the desired solution.

3.1.5 Linear Minimum Mean Square Error (LMMSE) Estimator

Let $\mathbf{x} \in \mathbb{R}^n$ a real Gaussian vector such that $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{W})$, where $\mathbf{W} \succ \mathbf{O}$. Consider the system model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}, \tag{3.50}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ a real matrix and $\mathbf{z} \in \mathbb{R}^m$ a Gaussian vector independent of $\mathbf{x}$, with $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{R})$ and $\mathbf{R} \succ \mathbf{O}$. Then, assuming that we only have access to the noisy measurements Eq. 3.50, the Linear Minimum Mean Square Error (LMMSE) estimator of $\mathbf{x}$ is given by [20], [16], [21]

$$\hat{\mathbf{x}} = \left(\mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{y}. \tag{3.51}$$

Similarly to the case of the linear system of equations of the previous Section, if we set $\mathbf{\Lambda} = \mathbf{W}^{-1}$, $\mathbf{\Xi} = \mathbf{A}$, $\mathbf{\Sigma} = \mathbf{R}^{-1}$, $\boldsymbol{\xi} = \mathbf{0}$ and $\mathbf{u} = \mathbf{y}$ in Eq. 3.30 we get the distribution

$$p(\mathbf{x}) = \mathcal{N}\left(\left(\mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{y}, \mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^{\top}\left(\mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right)\mathbf{x} + \mathbf{y}^{\top}\left(\mathbf{R}^{-1}\right)^{\top}\mathbf{A}\mathbf{x}\right\}. \tag{3.52}$$

Utilizing Gaussian Belief Propagation, we can infer the expected value of $p(\mathbf{x})$, which will ultimately yield the desired estimate $\hat{\mathbf{x}}$.

3.2 Average Consensus

Decentralized consensus algorithms involve a large number of agents that, using a distributed message passing approach, reach a converged agreement value [22], [23]. One of them is the so called *Average Consensus* algorithm. As the name suggests, this algorithm's agreement value is the average of all the initial values of the agents. In other words, after the algorithm converges, all agents have converged to the same value which is the average of all the initial ones.

In particular, the algorithm's iterations in matrix form are given by

$$\mathbf{x}^{(l)} = \mathbf{W}\mathbf{x}^{(l\ \ 1)}, \qquad l = 1, 2, \ldots, \tag{3.53}$$

where $\mathbf{W}$ is a weight matrix that encapsulates both the connectivity and the weights of the network. It can be shown that Eq. 3.53 reaches the fixed point

$$\mathbf{x}\ \ = \lim_{l\ !\ +1} \mathbf{x}^{(l)} = \lim_{l\ !\ +1} \mathbf{W}^{l}\mathbf{x}^{(0)} = \frac{\mathbf{1}\mathbf{1}^{>}}{n}\mathbf{x}^{(0)} = \frac{\mathbf{1}}{n}\sum_{i=1}^{n} x_i^{(0)}, \tag{3.54}$$

or equivalently

$$\lim_{l\ !\ +1} \mathbf{W}^{l} = \frac{\mathbf{1}\mathbf{1}^{>}}{n}, \tag{3.55}$$

where $\mathbf{1}$ denotes the n-nth dimensional vector whose all entries are 1, if and only if $\mathbf{1}^{>}\mathbf{W} = \mathbf{1}^{>}$, $\mathbf{W}^{>}\mathbf{1} = \mathbf{1}$ and $\rho\left(\mathbf{W} \quad \frac{\mathbf{1}\mathbf{1}^{>}}{n}\right) < 1$ [24].

Let $G = (V, E)$ the graph that stems from the topology of the network of agents, where V its vertices and E its edges. Then, the matrices that contain the *max-degree* weights of G,

$$\mathbf{W}_{ij} = \begin{cases} \dfrac{1}{d+1}, & i \ 6\!\!= j, fi, jg\ 2\ E, \\ 1 \quad \dfrac{d_i}{d+1}, & i = j, \\ 0, & i \ 6\!\!= j, fi, jg \not\!\!2\ E, \end{cases} \tag{3.56}$$

where d is the degree of G and d_i the degree of vertex i, is a family of matrices that satisfy the conditions above and hence guarantee convergence [25]. On the other hand, it is straightforward to observe that Eq. 3.53 falls into the system model that we discussed in Chapter 2, with $\mathbf{A} = \mathbf{W}$ and $\mathbf{b} = \mathbf{0}$. As a result, its asynchronous variant

$$\mathbf{x}^{(l)} = \left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} \quad \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l\ \ 1)} \tag{3.57}$$

can be considered.

Chapter 4

Convergence Analysis

In this Chapter, we provide some results regarding the convergence of the linear fixed point problem, for both the cases of synchronous and asynchronous scheduling.

4.1 General results for the synchronous case

Assume the *synchronous* linear update of Eq. 2.1. It is straightforward to see that

$$\begin{aligned}\mathbf{x}^{(l)} &= \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b} \\ &= \mathbf{A}\left[\mathbf{A}\mathbf{x}^{(l-2)} + \mathbf{b}\right] + \mathbf{b} \\ &= \mathbf{A}^2\mathbf{x}^{(l-2)} + (\mathbf{I} + \mathbf{A})\,\mathbf{b} \\ &= \mathbf{A}^2\left[\mathbf{A}\mathbf{x}^{(l-3)} + \mathbf{b}\right] + (\mathbf{I} + \mathbf{A})\,\mathbf{b} \\ &= \mathbf{A}^3\mathbf{x}^{(l-3)} + \left(\mathbf{I} + \mathbf{A} + \mathbf{A}^2\right)\mathbf{b}.\end{aligned} \tag{4.1}$$

Therefore, if we continue the calculations recursively, we can show that

$$\begin{aligned}\mathbf{x}^{(l)} &= \mathbf{A}^l\mathbf{x}^{(0)} + \sum_{j=1}^{l}\mathbf{A}^{l-j}\mathbf{b} \\ &= \mathbf{A}^l\mathbf{x}^{(0)} + \left(\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{l-1}\right)\mathbf{b}.\end{aligned} \tag{4.2}$$

As a result, the existence and calculation of the fixed point of Eq. 2.1, $\mathbf{x}^* = \lim_{l\to+\infty}\mathbf{x}^{(l)}$, is based on the existence of the limits

$$\lim_{l\to+\infty}\mathbf{A}^l \text{ and } \lim_{l\to+\infty}\left(\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{l-1}\right). \tag{4.3}$$

In the literature, the series of matrices $\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \ldots$ is called *Neumann Series* and is the matrix generalization of the power series. According to [26], the following theorems hold.

Theorem 1. For $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\lim_{l \to +\infty} \mathbf{A}^l = \mathbf{O}$ if and only if $\rho(\mathbf{A}) < 1$.

Theorem 2. For $\mathbf{A} \in \mathbb{R}^{n \times n}$, the following three statements are equivalent.

The Neumann series $\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \dots$ converges.
$\rho(\mathbf{A}) < 1$.
$\lim_{l \to +\infty} \mathbf{A}^l = \mathbf{O}$.

In such a case, $(\mathbf{I} - \mathbf{A})^{-1}$ exists and

$$\sum_{l=0}^{+\infty} \mathbf{A}^l = (\mathbf{I} - \mathbf{A})^{-1}.$$

As a result, based on Eq. 4.2 and the Theorems above, the fixed point of Eq. 2.1 exists if and only if $\rho(\mathbf{A}) < 1$, in which case it holds that

$$\begin{aligned} \mathbf{x} &= \lim_{l \to +\infty} \mathbf{x}^{(l)} \\ &= \lim_{l \to +\infty} \left[\mathbf{A}^l \mathbf{x}^{(0)} + \left(\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \dots + \mathbf{A}^{l-1} \right) \mathbf{b} \right] \\ &= \lim_{l \to +\infty} \mathbf{A}^l \mathbf{x}^{(0)} + \lim_{l \to +\infty} \left(\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \dots + \mathbf{A}^{l-1} \right) \mathbf{b} \\ &= (\mathbf{I} - \mathbf{A})^{-1} \mathbf{b}. \end{aligned} \tag{4.4}$$

Now, consider a different scenario, where we assume that in particular, $\mathbf{b} = \mathbf{0}$ in Eq. 2.1; namely the linear update is homogenous. In that case, Eq. 4.2 becomes

$$\mathbf{x}^{(l)} = \mathbf{A}^l \mathbf{x}^{(0)}. \tag{4.5}$$

As we can see, the term that corresponds to the Neumann series is not existent in the particular case of $\mathbf{b} = \mathbf{0}$. As a result, the convergence conditions are slightly different. In that case, we observe that the fixed point is now only dependent on the $\lim_{l \to +\infty} \mathbf{A}^l$, a limit that, in contrast to the case where $\mathbf{b} \neq \mathbf{0}$, may not be equal to the zero matrix $\mathbf{O}$. In order to discuss this, we firstly have to introduce some definitions from [26].

Definition 1. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$. The *spectrum* of $\mathbf{A}$, denoted by $\sigma(\mathbf{A})$, corresponds to the set of the *distinct* eigenvalues of $\mathbf{A}$, namely $\sigma(\mathbf{A}) = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$.

Definition 2. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\lambda \in \sigma(\mathbf{A}) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$.

- The *algebraic multiplicity* of λ is the number of times it is repeated as a root of the characteristic polynomial $\det(s\mathbf{I} - \mathbf{A})$.
- The *geometric multiplicity* of λ is equal to $\dim N(\mathbf{A} - \lambda\mathbf{I})$. In other words, it is the maximal number of linearly independent eigenvectors associated with λ.
- If the *algebraic* multiplicity of λ is equal to the *geometric* multiplicity of λ, then λ is called a *semisimple eigenvalue* of $\mathbf{A}$.

In addition, the following theorem holds [26].

Theorem 3. For $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\lim_{l \to \infty} \mathbf{A}^l$ exists if and only if

$\rho(\mathbf{A}) < 1$, or else
$\rho(\mathbf{A}) = 1$, where $\lambda = 1$ is the only eigenvalue on the unit circle, and $\lambda = 1$ is semisimple.

In both cases, when it exists, it holds that

$$\lim_{l \to \infty} \mathbf{A}^l = \text{the projector onto } N(\mathbf{I} - \mathbf{A}) \text{ along } R(\mathbf{I} - \mathbf{A}).$$

As a result, according to Theorem 3, we conclude that when $\mathbf{b} = \mathbf{0}$, the fixed point of the linear iterations of Eq. 2.1,

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)}, \tag{4.6}$$

exists if and only if $\rho(\mathbf{A}) < 1$ or $\rho(\mathbf{A}) = 1$, where $\lambda = 1$ is semisimple and the only eigenvalue with unitary magnitude, and it is equal to the projector onto $N(\mathbf{I} - \mathbf{A})$ along $R(\mathbf{I} - \mathbf{A})$.

4.2 Results in special cases of asynchronous scheduling

4.2.1 Gaussian Belief Propagation

Consider Gaussian Belief Propagation, as we discussed it in Section 3.1.3. We will prove the following.

Theorem 4 ([13])**.** With high-order decomposition and $\mathbf{v}^{(0)}$ *2* Θ, the expectation of Belief Propagation message mean $\mathsf{E}[\boldsymbol{\mu}^{(l)}]$ converges under the stochastic asynchronous scheduling that we presented in Chapter 2 if and only if $\rho\left(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I}\right) < 1$, where $\mathsf{E}[\boldsymbol{\Psi}^{(l)}]$, $\mathbf{P}$, *8*(l).

Proof. As we derived in Section 3.1.3, the Belief Propagation message means are updated linearly according to the iteration

$$\boldsymbol{\mu}^{(l)} = \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right)\boldsymbol{\mu}^{(l \ \ 1)} + \boldsymbol{\Psi}^{(l)}\mathbf{c}, \tag{4.7}$$

where $\mathbf{A}$ and $\mathbf{c}$ are known. The fixed point of Eq. 4.7, $\boldsymbol{\mu}$, must satisfy [1]

$$\boldsymbol{\mu} \ = \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right)\boldsymbol{\mu} \ + \boldsymbol{\Psi}^{(l)}\mathbf{c}. \tag{4.8}$$

Hence, it holds that

$$\boldsymbol{\mu}^{(l)} \quad \boldsymbol{\mu} \ = \mathbf{B}^{(l)}\left(\boldsymbol{\mu}^{(l \ \ 1)} \quad \boldsymbol{\mu} \ \right), \tag{4.9}$$

where $\mathbf{B}^{(l)}$, $\left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right)$. As a result,

$$\begin{aligned} \mathsf{E}[\boldsymbol{\mu}^{(l)} \quad \boldsymbol{\mu} \ ] &= \mathsf{E}\left[\mathbf{B}^{(l)}\left(\boldsymbol{\mu}^{(l \ \ 1)} \quad \boldsymbol{\mu} \ \right)\right] \\ &= \mathsf{E}[\mathbf{B}^{(l)}]\mathsf{E}\left[\left(\boldsymbol{\mu}^{(l \ \ 1)} \quad \boldsymbol{\mu} \ \right)\right], \end{aligned} \tag{4.10}$$

due to statistical independence. As we can see, the update of Eq. 4.10 is a linear equation. Thus, $\mathsf{E}[\boldsymbol{\mu}^{(l)} \quad \boldsymbol{\mu} \ ]$ converges to zero if and only if $\rho\left(\mathsf{E}[\mathbf{B}^{(l)}]\right) = \rho\left(\mathsf{E}\left[\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right]\right) < 1$. Therefore, if we assume that $\boldsymbol{\mu}$ is constant, we conclude that $\mathsf{E}[\boldsymbol{\mu}^{(l)}]$ converges to $\boldsymbol{\mu}$ if and only if $\rho\left(\mathsf{E}\left[\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} \quad \boldsymbol{\Psi}^{(l)}\right]\right) = \rho\left(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I}\right) < 1$.

4.2.2 Average Consensus

We firstly introduce what convergence *in expectation* is [27], [28].

Definition 3. A sequence of random vectors $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ converges to $\mathbf{x}$ *in expectation*, if $\lim_{l \ ! \ +\ 1} \mathsf{E}[\mathbf{x}^{(l)}] = \mathbf{x}$.

In this work, we derive the following Theorem.

[1] Eq. 4.8 holds if we assume that $\boldsymbol{\mu}$ is independent of *all* the scheduling matrices $\boldsymbol{\Psi}^{(l)}$.

Theorem 5. Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ be a rectangular stochastic matrix, namely

$$\mathbf{1}^{\top}\mathbf{W} = \mathbf{1}^{\top} \tag{4.11}$$

$$\mathbf{W}^{\top}\mathbf{1} = \mathbf{1}. \tag{4.12}$$

Suppose in addition that $\mathbb{E}[\mathbf{\Psi}^{(l)}] = p\mathbf{I}$, for $p \in (0,1]$ and $\forall l$. Then, the asynchronous variant of Average Consensus in Eq. 3.57 converges *in expectation* to the Average Consensus fixed point $\mathbf{x}^{*} = \frac{\mathbf{1}\mathbf{1}^{\top}}{n}\mathbf{x}^{(0)}$ if

$$\rho\left(p\mathbf{W} + (1-p)\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^{\top}}{n}\right) < 1.$$

Proof. Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ satisfy 4.11 and 4.12 and $\mathbf{B}^{(l)} = \mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} - \mathbf{\Psi}^{(l)}$, where $\mathbf{\Psi}^{(l)}$ the scheduling matrix as described in Chapter 2. Then, if $\mathbf{P} = \mathbb{E}[\mathbf{\Psi}^{(l)}] = p\mathbf{I}$, $p \in (0,1]$, the matrix $\mathbb{E}[\mathbf{B}^{(l)}] = p\mathbf{W} + (1-p)\mathbf{I}$ also satisfies the conditions above, since

$$\mathbf{1}^{\top}(p\mathbf{W} + (1-p)\mathbf{I}) = p\mathbf{1}^{\top}\mathbf{W} + (1-p)\mathbf{1}^{\top}\mathbf{I} = \mathbf{1}^{\top} \tag{4.13}$$

$$(p\mathbf{W} + (1-p)\mathbf{I})^{\top}\mathbf{1} = p\mathbf{W}^{\top}\mathbf{1} + (1-p)\mathbf{I}^{\top}\mathbf{1} = \mathbf{1}. \tag{4.14}$$

As a result, consider the asynchronous variant of Average Consensus in Eq. 3.57,

$$\mathbf{x}^{(l)} = \left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)}. \tag{4.15}$$

Then, it follows that

$$\begin{aligned}\mathbb{E}[\mathbf{x}^{(l)}] &= \mathbb{E}[\left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)}] \\ &= \mathbb{E}[\left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)]\mathbb{E}[\mathbf{x}^{(l-1)}] \\ &= (\mathbf{P}\mathbf{W} + \mathbf{I} - \mathbf{P})\,\mathbb{E}[\mathbf{x}^{(l-1)}],\end{aligned} \tag{4.16}$$

since at iteration (l), $\mathbf{\Psi}^{(l)}$ and $\mathbf{x}^{(l-1)}$ are statistically independent. Then, since $\mathbb{E}[\mathbf{\Psi}^{(l)}] = p\mathbf{I}$, $p \in (0,1]$, $\forall l$, we get

$$\mathbb{E}[\mathbf{x}^{(l)}] = (p\mathbf{W} + (1-p)\mathbf{I})\,\mathbb{E}[\mathbf{x}^{(l-1)}]. \tag{4.17}$$

Therefore, if the statistics of the independent matrices $\mathbf{\Psi}^{(l)}$ result in $\rho\left(p\mathbf{W} + (1-p)\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^{\top}}{n}\right) < 1$, then the iteration Eq. 4.15 converges *in expectation* to the Average Consensus fixed point $\mathbf{x}^{*} = \frac{\mathbf{1}\mathbf{1}^{\top}}{n}\mathbf{x}^{(0)}$ [28].

We will now present a result regarding the convergence rate of Average Consensus from relevant work in [27]. In particular, in this work it is additionally assumed that all $\mathbf{B}^{(l)}$ in the average consensus iterations

$$\mathbf{x}^{(l)} = \mathbf{B}^{(l)}\mathbf{x}^{(l-1)} \tag{4.18}$$

are symmetric and stochastic matrices, namely $\mathbf{1}^\top \mathbf{B}^{(l)} = \mathbf{1}^\top$ and $\mathbf{B}^{(l)\top}\mathbf{1} = \mathbf{1}$, $\forall l$. Hence, we introduce the following definition.

Definition 4 ((α,γ)-convergence Time [27]). For a sequence of random vectors $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$, given any $\gamma \in [0,1]$ and $\alpha \in \mathbb{R}_+$, suppose that $\|\mathbf{x}^{(0)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}\| \neq 0$, where $\|\cdot\|$ some norm. Then, the sequence's (α,γ)-convergence time T_k is defined as

$$T_k = \inf\left\{ k : \Pr\left(\frac{\|\mathbf{x}^{(k)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}\|}{\|\mathbf{x}^{(0)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}\|} \leq \alpha \right) \geq \gamma \right\},$$

and indicates the least iterations needed for $\mathbf{x}^{(k)}$ to be α close to $\frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}$ with probability at least γ.

Since $\mathbf{B}^{(l)}$ are stochastic matrices, it holds that

$$\begin{aligned} \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(l)} &= \frac{\mathbf{1}\mathbf{1}^\top}{n}\prod_{k=1}^{l}\mathbf{B}^{(l-k+1)}x^{(0)} \\ &= \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}. \end{aligned} \tag{4.19}$$

Therefore, we get

$$\begin{aligned} \mathbf{x}^{(l)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)} &= \mathbf{B}^{(l)}\mathbf{x}^{(l-1)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)} \\ &\overset{4.19}{=} \left(\mathbf{B}^{(l)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\mathbf{x}^{(l-1)} \\ &= \prod_{k=1}^{l}\left(\mathbf{B}^{(l-k+1)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\mathbf{x}^{(0)} \end{aligned} \tag{4.20}$$

and as a result

$$\mathbb{E}\left[\mathbf{x}^{(l)} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)}\right] = \prod_{k=1}^{l}\left(\mathbb{E}\left[\mathbf{B}^{(l-k+1)}\right] - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\mathbf{x}^{(0)}, \tag{4.21}$$

where $\mathbb{E}\left[\mathbf{B}^{(l-k+1)}\right]$ is constant across all the independent iteration and known. Hence, the following Theorem holds.

Theorem 6 ([27]). If $\rho(C) < 1$ and $\rho(\mathbb{E}\left[\mathbf{B}^{(l-k+1)}\right] - \frac{\mathbf{1}\mathbf{1}^\top}{n}) < 1$, then the (α,γ)-convergence time T_k of Average Consensus is upper bounded by

$$T_{ku} = \left\lceil \frac{\log\left(\alpha^2(1-\gamma)\right)}{\log \rho(C)} \right\rceil$$

and there exists an $\mathbf{x}^{(0)}$ such that T_k is lower bounded by

$$T_{kl} = \left\lfloor \frac{\log(1 \quad \gamma + \gamma\alpha^2)}{2\log\rho\left(\mathsf{E}\left[\mathbf{B}^{(l \quad k+1)}\right] \quad \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)} \right\rfloor,$$

where $C = \mathsf{E}\left[\left(\mathbf{B}^{(l)} \quad \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)^\top \left(\mathbf{B}^{(l)} \quad \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\right]$

Proof. [27].

Although those results are not directly applicable to our framework, since in our case, the matrices $\mathbf{\Psi}^{(l)}\mathbf{W} \quad \mathbf{I} + \mathbf{\Psi}^{(l)}$ are *rarely* column stochastic, they give us a better insight on not only how to evaluate the efficiency of algorithms like Average Consensus, but also on the parameters that affect their convergence speed the most; as expected, the statistics of the random matrices $\mathbf{B}^{(l)}$ play the greatest role.

Chapter 5

Simulations

In this Chapter, we present the experimental results of the *asynchronous* variants of Gaussian Belief Propagation and Average Consensus. In particular, we provide simulations studying the convergence of the aforementioned algorithms and their comparison with their synchronous cases.

5.1 Gaussian Belief Propagation

5.1.1 Linear System Solution

In this Section we present numerical results regarding the convergence of Gaussian Belief Propagation with synchronous and asynchronous schedulings. Following Section 3.1.4, our task is to solve the linear system of equations

$$\mathbf{M}\mathbf{x} = \mathbf{s}. \tag{5.1}$$

In all the experiments, we assume that [13]

$$\mathbf{M} = \begin{bmatrix} 3.63 & 6.12 & 0 & 0 & 0 & 2.61 & 0 & 0 \\ 0 & 10.65 & 7.59 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.92 & 7.05 & 0 & 0 & 10.46 & 0 \\ 0 & 0 & 0 & 0.18 & 3.27 & 0 & 0 & 0 \\ 2.01 & 0 & 0 & 0 & 0.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8.01 & 0.37 \\ 5.18 & 1.86 & 0 & 4.63 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.91 & 0 & 0 & 0 & 0 & 0.13 \end{bmatrix} \tag{5.2}$$

and

$$\mathbf{s} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{>}. \tag{5.3}$$

As we described in the aforementioned Section, it is possible to find the solution of Eq. 5.1,

$$\mathbf{x} = \left(\mathbf{M}^{>}\mathbf{M}\right)^{1}\mathbf{M}^{>}\mathbf{s}, \tag{5.4}$$

inferring the expected value of the distribution

$$p(\mathbf{x}) \ / \ \exp\left\{ \ \frac{1}{2}\mathbf{x}^{>}\mathbf{M}^{>}\mathbf{M}\mathbf{x} + \mathbf{s}^{>}\mathbf{M}\mathbf{x} \right\}, \tag{5.5}$$

using Gaussian Belief Propagation.

The *loopy* factor graph that is associated with the given $\mathbf{M}$ and $\mathbf{s}$ and represents the distribution function $p(\mathbf{x})$ can be seen in Fig. 5.1. As we described in Section 3.1.3, at iteration (l), the message means $\mu_{g_j \ ! \ x_i}$ of Gaussian Belief Propagation are updated *linearly*, according to Eq. 3.43. Therefore, our goal is to assign different *edges* of the factor graph to different WSN nodes, in order to distribute the computations, according to the system model that we discussed in Chapter 2.

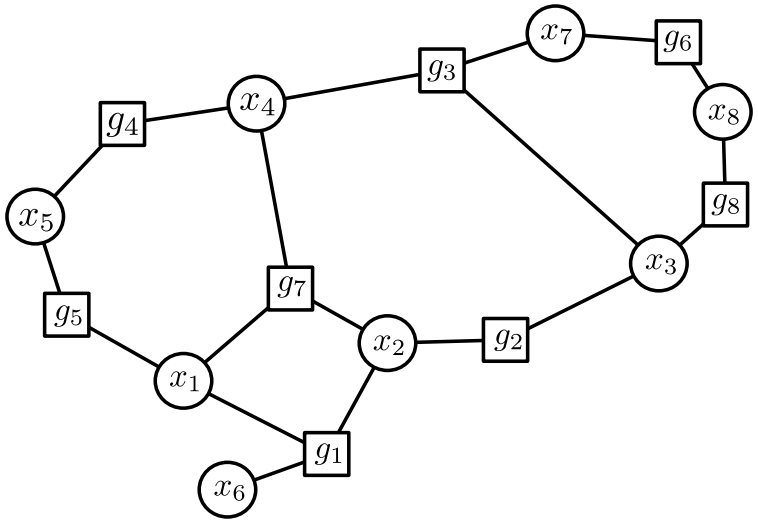


Figure 5.1: The factor graph that corresponds to $p(\mathbf{x})$ in Eq. 5.5, for $\mathbf{M}$ and $\mathbf{s}$ that are given in Eq. 5.2 and 5.3 , respectively. Note that factors f_i are not depicted, for the sake of simplicity.

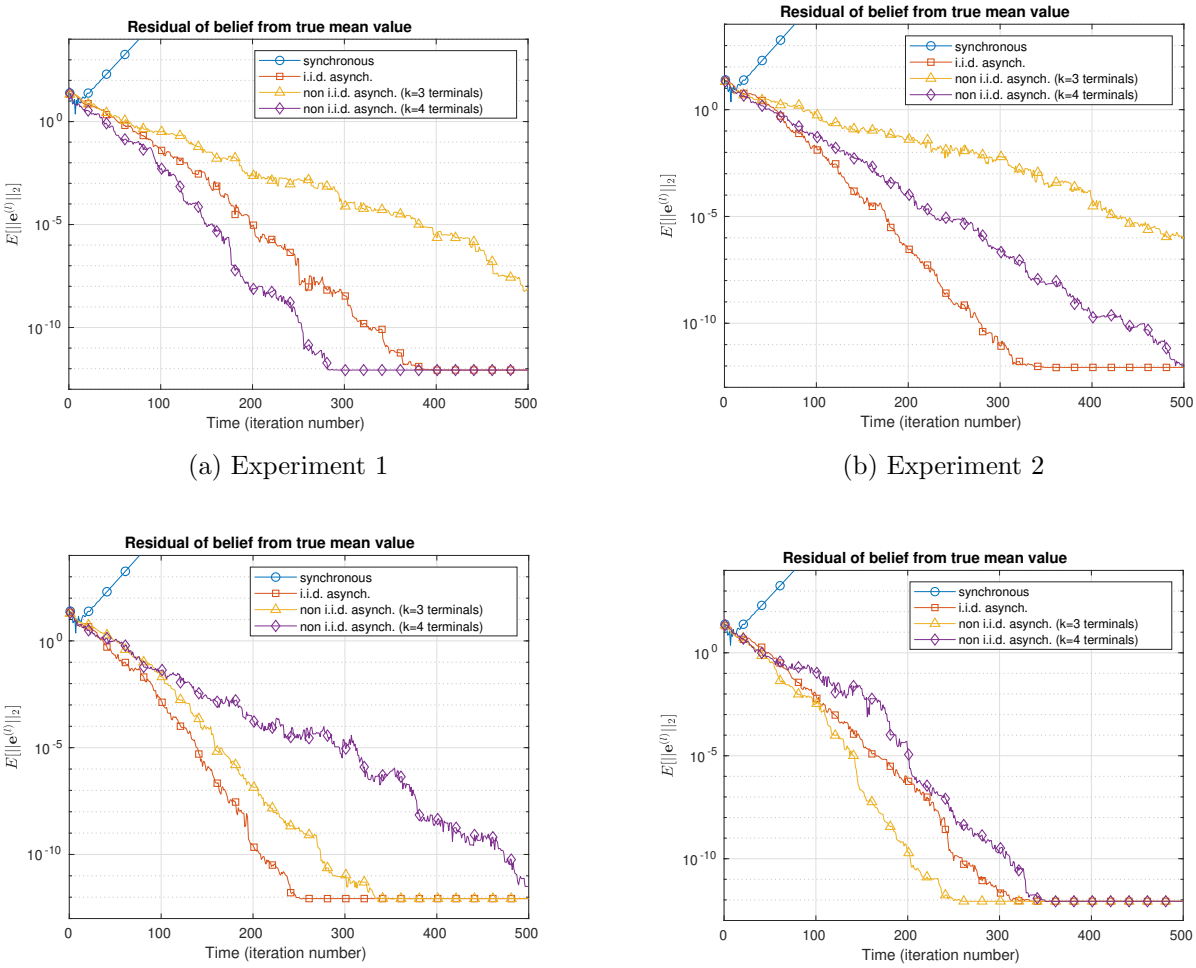
In all the experiments, we consider *three* different kinds of schedules:

1. A *synchronous scheduling*, where the whole vector $\boldsymbol{\mu}$ is assigned to a single WSN node, that is assumed to always have sufficient energy to operate. As a result, at each iterations (l) of Gaussian Belief Propagation, *all* elements of $\boldsymbol{\mu}^{(l \ \ 1)}$ are updated.

2. An *i.i.d. asynchronous scheduling* [13], where essentially *every* element of $\boldsymbol{\mu}$ is assigned to a *di erent* WSN terminal. On top of that, the scheduling variables $\psi_i^{(l)}$ (refer to Eq. 2.3) are assumed i.i.d. Bernoulli random variables, with the same parameter p. In other words, the probability that at iteration (l), a WSN node will have sufficient energy in order to update the corresponding element of $\boldsymbol{\mu}$ is p and is independent of all the other terminals or previous iterations.

3. A *non i.i.d. asynchronous scheduling*, where $\boldsymbol{\mu}$ is partitioned into k *distinct* WSN terminals. Each terminal is responsible for the update of the elements that it encloses, and as a result, its energy sufficiency affects their update. In contrast to the previous case, the scheduling variables $\psi_i^{(l)}$ are assumed Bernoulli random variables, all independent across different iterations,

but possibly *dependent* across the different elements of $\boldsymbol{\mu}$. More specifically, we assume that there exist k distinct scheduling variables; the elements of $\boldsymbol{\mu}$ that are part of the same WSN node are also assigned with the *same* scheduling variables. Hence, all $\psi_t^{(l)}$ that correspond to $\mu_{g_j \,!\, x_i}^{(l)}$ which belong to the same WSN terminal, are *fully dependent* and in fact exactly the same. To put it differently, at each iteration, k independent -and possibly *non* identically distributed- Bernoulli random variables (each for every node) are sampled, and -according to their result- the elements that correspond to each WSN terminal are updated.



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

(d) Experiment 4

Figure 5.2: Convergence of asynchronous Gaussian Belief Propagation for different experiments, in the case of solving a linear system of equations. In particular, we present results for WSNs with different number of terminals, different vector-to-terminal assignments, different outage probabilities and different schedulings.

In Fig. 5.2, we present 4 different experimental results for the aforementioned framework. In particular, we plot the estimation of the per iteration expected value $\mathsf{E}[k\mathbf{e}^{(l)}k_2]$ using 20 independent

experiments; at iteration (l), the error $\mathbf{e}$ is defined as

$$\mathbf{e}^{(l)} \quad , \quad \boldsymbol{\epsilon}^{(l)} \quad \hat{\mathbf{x}}, \tag{5.6}$$

where $\boldsymbol{\epsilon}^{(l)}$ is the vector that is constructed if we stack all *belief means*, according to Eq. 3.41, and $\hat{\mathbf{x}}$, $\left(\mathbf{M}^{\top}\mathbf{M}\right)^{-1}\mathbf{M}^{\top}\mathbf{s}$, the least squares solution of the system Eq. 5.1. Moreover, particularly for the case of the non i.i.d. asynchronous scheduling that we previously discussed, in each experiment we considered two cases where the WSN that we simulate has $k = 3$ and $k = 4$ terminals, respectively. In addition, the different elements of $\boldsymbol{\mu}$ where randomly assigned to each WSN terminal in every case. Finally, in Tables 5.1, 5.2, 5.3 and 5.4, we present the parameters that were used in each experiment, along with the necessary convergence metrics. Note that with p_i we denote the probability of the successful update of the elements that are assigned to WSN terminal i. Also notice that in Experiment 4, we assume that *all* the involved WSN terminals are identical, in the sense that the probability that their enclosed values are updated at iteration (l) is constant for all of them.

Table 5.1: Experiment 1.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	**1.02231**	**1.02231**
i.i.d. asynchronous	$p = 0.4025$	**1.02231**	0.89954
non i.i.d. asynchronous $(k = 3)$	$p_1 = 0.3576$ $p_2 = 0.3453$ $p_3 = 0.2699$	**1.02231**	0.9153
non i.i.d. asynchronous $(k = 4)$	$p_1 = 0.8355$ $p_2 = 0.7947$ $p_3 = 0.4941$ $p_4 = 0.8390$	**1.02231**	0.92293

Table 5.2: Experiment 2.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	**1.02231**	**1.02231**
i.i.d. asynchronous	$p = 0.4709$	**1.02231**	0.88245
non i.i.d. asynchronous $(k = 3)$	$p_1 = 0.7343$ $p_2 = 0.7147$ $p_3 = 0.4417$	**1.02231**	0.86944
non i.i.d. asynchronous $(k = 4)$	$p_1 = 0.4958$ $p_2 = 0.8295$ $p_3 = 0.7512$ $p_4 = 0.4456$	**1.02231**	0.84002

Taking a closer look at the aforementioned Figures and Tables, one can make some useful observations. First of all, we can see that $\rho(\mathbf{A}) = 1.02231 > 1$. Therefore, as we derived in Chapter 4, we expect that the linear iterations of Eq. 3.43 diverges [29], something that one can verify

Table 5.3: Experiment 3.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	**1.02231**	**1.02231**
i.i.d. asynchronous	$p = 0.5922$	**1.02231**	0.85218
non i.i.d. asynchronous ($k = 3$)	$p_1 = 0.6503$ $p_2 = 0.2540$ $p_3 = 0.8426$	**1.02231**	0.87051
non i.i.d. asynchronous ($k = 4$)	$p_1 = 0.6271$ $p_2 = 0.5314$ $p_3 = 0.7938$ $p_4 = 0.8454$	**1.02231**	0.85743

Table 5.4: Experiment 4.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	**1.02231**	**1.02231**
i.i.d. asynchronous	$p = 0.5122$	**1.02231**	0.87214
non i.i.d. asynchronous ($k = 3$)	$p_1 = 0.5122$ $p_2 = 0.5122$ $p_3 = 0.5122$	**1.02231**	0.87214
non i.i.d. asynchronous ($k = 4$)	$p_1 = 0.5122$ $p_2 = 0.5122$ $p_3 = 0.5122$ $p_4 = 0.5122$	**1.02231**	0.87214

in Fig. 5.2. In addition, we should predict that all the different asynchronous schedulings should converge, since $\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I}) < 1$ in all the cases, something that is also experimentally proved in the simulations.

To continue, we see that choosing different number of WSN terminals k, different probabilities p_i and different partitions of the calculated vector can *heavily* change the behaviour of the algorithm. For instance, notice how utilizing an non i.i.d. asynchronous scheduling with $k = 4$ WSN terminals becomes from the fastest in Experiment 1 to the slowest possible scheduling in Experiment 3, when the involved parameters are altered. Moreover, taking a thorough look at the Figure of Experiment 4, we see that even if the energy statistics of each WSN terminal are the same, the difference in the number of the aforementioned nodes along with the different vector partitioning can heavily impact the convergence speed.

Besides all the above -however-, perhaps the most important conclusion that one would make is that in this particular instance, the unreliability of the WSN terminals is not -really- a limitation. In contrast, it becomes a key *feature*, since in a different scenario when the WSN would allow synchronous calculations, Gaussian Belief Propagation would diverge.

5.1.2 Linear Minimum Mean Square Error (LMMSE) Estimator

In the next experiments, our goal is to calculate the Linear Minimum Mean Square Error (LMMSE) estimator of a Gaussian vector. More specifically, let $\mathsf{R}^n \ni \mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{W})$ and $\mathsf{R}^m \ni \mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{R})$, where $\mathbf{W} \neq \mathbf{O}$ and $\mathbf{R} \neq \mathbf{O}$. Then, assuming the linear model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}, \tag{5.7}$$

where $\mathbf{A} \in \mathsf{R}^{m \times n}$, the LMMSE estimator of $\mathbf{x}$,

$$\hat{\mathbf{x}} = \left(\mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{y}, \tag{5.8}$$

can be calculated by inferring the expected value of the distribution

$$p(\mathbf{x}) = \mathcal{Z}^{-1} \exp\left\{-\frac{1}{2}\mathbf{x}^{\top}\left(\mathbf{W}^{-1} + \mathbf{A}^{\top}\mathbf{R}^{-1}\mathbf{A}\right)\mathbf{x} + \mathbf{y}^{\top}\left(\mathbf{R}^{-1}\right)^{\top}\mathbf{A}\mathbf{x}\right\}. \tag{5.9}$$

using Gaussian Belief Propagation, as described in Section 3.1.5,

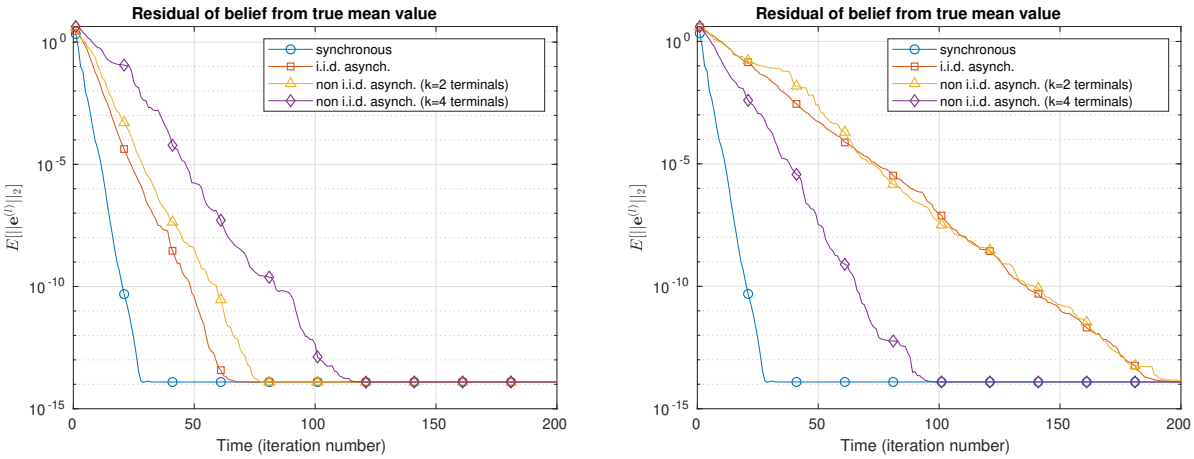
In our experiments, we assume that $\mathbf{W} = \mathbf{I}$, $\mathbf{R} = 0.1\mathbf{I}$,

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.411 \\ 1.709 & 0 & 0 & 0 & 0 & 0.084 & 0 & 0.876 \\ 0.399 & 0 & 0.439 & 1.753 & 0 & 0 & 0 & 0.232 \\ 0 & 0.641 & 0 & 0 & 1.679 & 0.266 & 0 & 1.869 \\ 0 & 0 & 1.365 & 0 & 0.508 & 0 & 0 & 3.006 \\ 0.458 & 0.320 & 0 & 0.141 & 0.651 & 0 & 0 & 0.299 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.379 \\ 0.261 & 0 & 0.546 & 0.205 & 0.174 & 0 & 0 & 0.328 \end{bmatrix}, \tag{5.10}$$

and we test the same schedulings as in the case of the Linear System in the previous section; especially for the case of non i.i.d. asynchronous scheduling, we test two different WSNs with $k = 2$ and $k = 4$ terminals, respectively, and different update probabilities. In addition, in order to assign the different parts of $\mathbf{x}$ to different WSN nodes, we utilize *k-means* algorithm, as we presented it in Chapter 2.

Table 5.5: Experiment 2.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \neq \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	0.5476	0.5476
i.i.d. asynchronous	$p = 0.7712$	0.5476	0.64932
non i.i.d. asynchronous ($k = 2$)	$p_1 = 0.7776$ $p_2 = 0.5088$	0.5476	0.71418
non i.i.d. asynchronous ($k = 4$)	$p_1 = 0.3415$ $p_2 = 0.8605$ $p_3 = 0.7642$ $p_4 = 0.4201$	0.5476	0.75346



(a) Experiment 1

(b) Experiment 2

Figure 5.3: Convergence of Gaussian Belief Propagation for calculating the LMMSE estimator of a Gaussian vector, under different WSNs and schedulings.

Table 5.6: Experiment 2.

Scheduling	Probability of update	$\rho(\mathbf{A})$	$\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I})$
synchronous	$p = 1$	0.5476	0.5476
i.i.d. asynchronous	$p = 0.3670$	0.5476	0.83215
non i.i.d. asynchronous ($k = 2$)	$p_1 = 0.3909$ $p_2 = 0.2286$	0.5476	0.86542
non i.i.d. asynchronous ($k = 4$)	$p_1 = 0.6049$ $p_2 = 0.7654$ $p_3 = 0.4822$ $p_4 = 0.8205$	0.5476	0.73958

In Figs. 5.3a, 5.3b and in Tables 5.5, 5.6 we show the convergence plots (similarly to the previous case, we plot the mean $\mathsf{E}[k\boldsymbol{\epsilon}^{(l)} \quad \hat{\mathbf{x}}k_2]$ at every iteration) along the parameters that were used in each experiment. First of all, we notice that $\rho(\mathbf{A}) = 0.5476 < 1$; as a result, in contrast to the previous set of experiments, we expect Gaussian Belief Propagation to converge under a synchronous scheduling, something we can verify from the plots. In addition, we see that $\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I}) < 1$ for all the different asynchronous schedulings, something that -as we can derive from the plots- implies convergence, in all the cases.

Thus, taking a closer look to the aforementioned parameters and plots, we draw similar conclusions to the case of solving a Linear System. More specifically, we see that when a vector is clustered with a different set of update probabilities and different number of WSN terminals k, the convergence speed at which asynchronous Gaussian Belief Propagation converges changes. Notice -for example- how an asynchronous scheduling with $k = 4$ changes from being the slowest to the fastest scheduling, along the asynchronous ones with different p's and k's, if we alter the update probabilities of the network. Therefore, it is particularly interesting to study the clustering methods

in greater depth, since they play a key role at the convergence rate.

5.2 Average Consensus

In this Section we provide numerical results regarding both the synchronous and asynchronous variants of the Average Consensus algorithm.

Firstly, we consider an example of a network with $n = 10$ agents. In particular, let

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0.4477 \\ 0.0278 \\ 1.3360 \\ 0.5996 \\ 0.9013 \\ 1.7324 \\ 0.8797 \\ 0.7707 \\ 0.6157 \\ 1.4005 \end{bmatrix}. \tag{5.11}$$

We then randomly generate an adjacency matrix for a graph with $n = 10$ nodes and generate the matrix $\mathbf{W}$ that contains its *max-degree* weights, as per Eq. 3.56

$$\mathbf{W} = \begin{bmatrix} 0.5000 & 0.1250 & 0 & 0 & 0 & 0.1250 & 0 & 0 & 0.1250 & 0.1250 \\ 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0 & 0.1250 & 0.1250 & 0 & 0.1250 \\ 0 & 0.1250 & 0.7500 & 0.1250 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1250 & 0.1250 & 0.5000 & 0 & 0.1250 & 0 & 0.1250 & 0 & 0 \\ 0 & 0.1250 & 0 & 0 & 0.7500 & 0.12500 & 0 & 0 & 0 & 0 \\ 0.1250 & 0 & 0 & 0.1250 & 0.1250 & 0.2500 & 0 & 0.1250 & 0.1250 & 0.1250 \\ 0 & 0.1250 & 0 & 0 & 0 & 0 & 0.7500 & 0.1250 & 0 & 0 \\ 0 & 0.1250 & 0 & 0.1250 & 0 & 0.1250 & 0.1250 & 0.5000 & 0 & 0 \\ 0.1250 & 0 & 0 & 0 & 0 & 0.1250 & 0 & 0 & 0.6250 & 0.1250 \\ 0.1250 & 0.1250 & 0 & 0 & 0 & 0.1250 & 0 & 0 & 0.1250 & 0.5000 \end{bmatrix}. \tag{5.12}$$

As expected, it holds that $\mathbf{1}^\top \mathbf{W} = \mathbf{1}^\top$, $\mathbf{W}^\top \mathbf{1} = \mathbf{1}$ and $\rho\left(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) = 0.8385 < 1$. Therefore, we would predict that the *synchronous* iteration of Eq. 3.53,

$$\mathbf{x}^{(l)} = \mathbf{W}\mathbf{x}^{(l-1)}, \tag{5.13}$$

reaches the fixed point

$$\mathbf{x}^* = \frac{\mathbf{1}\mathbf{1}^\top}{n}\mathbf{x}^{(0)} = 0.3547\,\mathbf{1}. \tag{5.14}$$

As a matter of fact, this prediction is verified, taking a look at Fig. 5.4. As we can see, after about 20 iterations, the synchronous linear iterations of Eq. 5.13 reach consensus and converge to a vector whose all entries are the true average of the elements of $\mathbf{x}^{(0)}$.

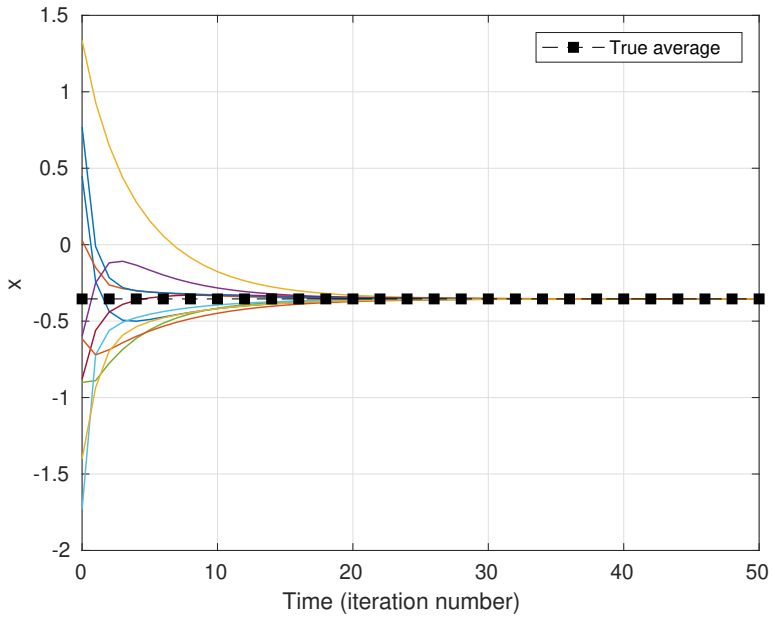


Figure 5.4: Convergence of Average Consensus with a synchronous scheduling, where $\mathbf{W}$ and $\mathbf{x}^{(0)}$ are given in Eqs. 5.12 and 5.11, respectively.

Now, we focus on the asynchronous variant of Average Consensus algorithm, given in Eq. 3.57, and below

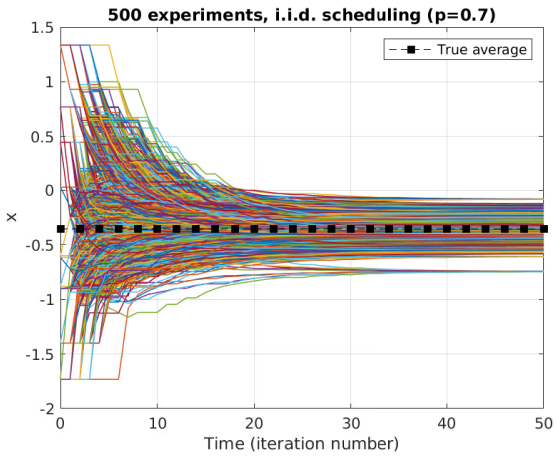
$$\mathbf{x}^{(l)} = \left(\mathbf{\Psi}^{(l)}\mathbf{W} + \mathbf{I} \quad \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l\ \ 1)}. \tag{5.15}$$

In particular, we study an i.i.d. asynchronous scheduling (as described in the previous Section), with $\mathsf{E}[\mathbf{\Psi}^{(l)}] = p\mathbf{I}$, $\forall l$, and $p = 0.7$. In Fig. 5.5a, we depict how the per iteration value of each agent behaves for 500 independent experiments. As we can see, in all the cases, the algorithm converges to an agreement which is not always identical with the true average. However, it holds that

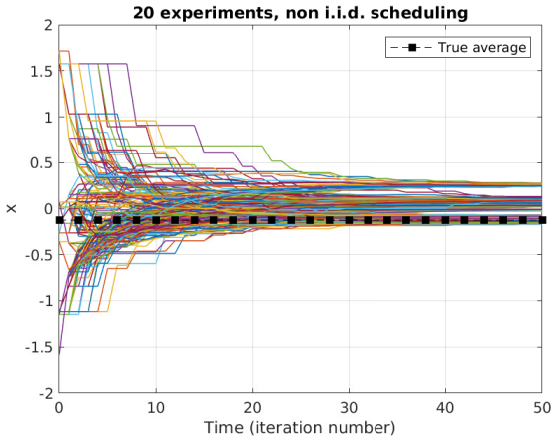
$$\rho\left(p(\mathbf{W} + (1 \quad p)\mathbf{I} \quad \frac{\mathbf{1}\mathbf{1}^{\top}}{n}\right) = 0.8869 < 1. \tag{5.16}$$

Therefore, since $\mathbf{W}$ satisfies the conditions 4.11, 4.12, as described in the Section 4.2.2, we should expect that Eq. 5.15 converges to $\frac{\mathbf{1}\mathbf{1}^{\top}}{n}\mathbf{x}^{(0)}$ *in expectation*. This is actually something that we verify, since the average of the converged values of the experiments presented in Fig. 5.5a yields the vector $\mathbf{x} = 0.3540\ \mathbf{1}$, which is fairly close to $\frac{\mathbf{1}\mathbf{1}^{\top}}{n}\mathbf{x}^{(0)} = 0.3547\ \mathbf{1}$.

To continue, in Fig. 5.5b we depict the behaviour of the algorithm when an asynchronous scheduling is utilized. More specifically, we assume that the entries of the diagonal matrix $\mathsf{E}[\mathbf{\Psi}^{(l)}] = \mathbf{P}$ are not identical. As a result, although the agents reach agreement in all experiments, the conditions that we described in Section 4.2.2 are not all satisfied. As a result, we should not expect *in expectation* convergence, something that we also observe in Fig. 5.5b, since the average of the converged values of the algorithm is considerable far from the true average.



(a) i.i.d. asynchronous scheduling



(b) non i.i.d. asynchronous scheduling

Figure 5.5: The behaviour of Average Consensus algorithm under: (a) an i.i.d. asynchronous scheduling, where $\mathsf{E}[\mathbf{\Psi}^{(l)}] = 0.7\,\mathbf{I}$, $\forall l$, (b) a non i.i.d. asynchronous scheduling.

Considering all the above, the most important conclusion that one could draw is that implementing Average Consensus in a distributed and partially reliable network is possible, if we insert *asynchrony* to the computation. However, since the agreement value may differ from the desired one, this comes at the expense of increased overall delay; in order to calculate an accurate estimate of the average of the vector, the algorithm must be repeated multiple times.

5.3 A "counterexample"

Let the matrix $\mathbf{A} \in \mathsf{R}^4$ with

$$\mathbf{A} = \begin{bmatrix} 1.1187 & 1.2697 & 1.3414 & 1.0726 \\ 0.5487 & 0.0458 & 0.5795 & 1.1289 \\ 0.0038 & 0.2531 & 0.3476 & 0.2019 \\ 0.5467 & 1.7122 & 0.4240 & 0.8038 \end{bmatrix}, \tag{5.17}$$

and assume that we want to solve the fixed point problem that we have discussed so far, utilizing both a synchronous and asynchronous scheduling (Eqs 2.1, 2.4). In other words, we want to find the fixed point of the iterations

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b} \tag{5.18}$$

and

$$\mathbf{x}^{(l)} = (\mathbf{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \mathbf{\Psi}^{(l)})\mathbf{x}^{(l-1)} + \mathbf{\Psi}^{(l)}\mathbf{b}, \tag{5.19}$$

respectively, where $\mathbf{b} \in \mathsf{R}^4$ and $\mathsf{E}[\mathbf{\Psi}^{(l)}] = 0.1\mathbf{I}$, $\forall l$.

We can verify that $\rho(\mathbf{A}) = 1.814246 > 1$ and $\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{I}) = 0.888649 < 1$. As a result, based on what we have discussed so far, one would expect that the synchronous iterations would diverge, whilst the asynchronous once would converge. Taking a look at Fig. 5.6, although the former is verified, we see that the linear iterations with an asynchronous scheduling diverge, despite the fact that the condition that we derived in Section 4.2.1 regarding Gaussian Belief Propagation is satisfied. As a result, we conclude that the aforementioned condition -probably- does not hold under the assumption that the fixed point of Eq. 5.19 is *not* independent of the scheduling matrices $\mathbf{\Psi}^{(l)}$. Therefore, the exact conditions under which Eq. 5.19 converge without this assumption of independence is a topic of research.

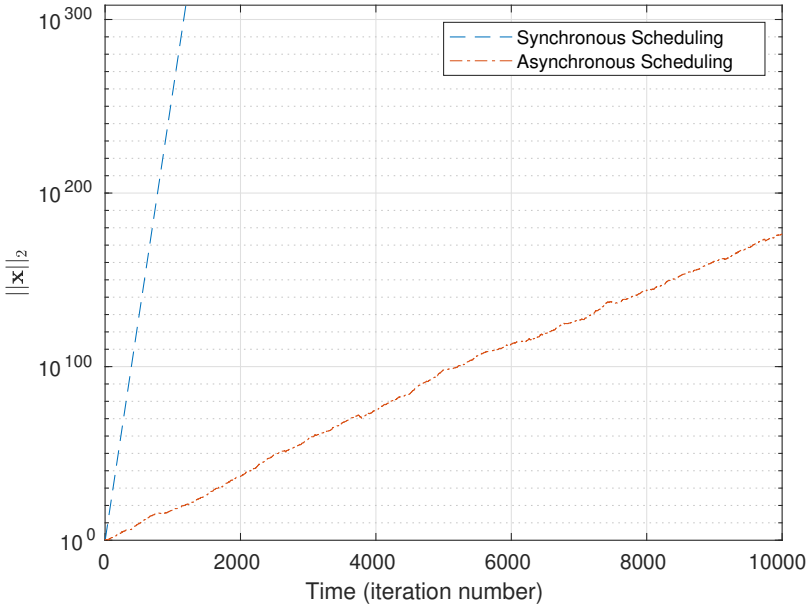


Figure 5.6: Behaviour of linear synchronous and asynchronous iterations for a case where $\rho(\mathbf{A}) > 1$ and $\rho(\mathbf{P}(\mathbf{A} \quad \mathbf{I}) + \mathbf{P}) < 1$.

Chapter 6

Conclusions and Future Work

In this work, our goal was to find the fixed point of

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l\ \ 1)} + \mathbf{b} \tag{6.1}$$

in a distributed manner, utilizing a Wireless Sensor Network (WSN) that is solely powered by the environment. In particular, since at some point of time the ambient power may not be sufficient for the computations, some nodes of the network may fail to operate, at some iterations. As a result, stochastic *asynchrony* is inserted to the computations, in the sense that at some iteration (l), some nodes of the WSN may fail to perform computations and hence, different elements of $\mathbf{x}^{(l\ \ 1)}$ that are assigned to different WSN nodes may (*randomly*) not be updated.

We considered two particularly famous inference algorithms that can be modeled with Eq. 6.1, *Gaussian Belief Propagation* and *Average Consensus*, and described some of their convergence properties, for both the cases of *synchronous* and *asynchronous* operations. Based on our analysis and providing some numerical results, we drew the following conclusions. Fistly, for the case of Gaussian Belief Propagation, we showed that *asynchrony* is not really a bug but a feature. In particular, we showed that it can determine whether Gaussian Belief Propagation will converge and that it plays a major role in its convergence rate. Afterwards, for the case of Average Consensus, we showed that if the stochastic *asynchrony* has a certain statistical property, then the algorithm converges *in expectation*. As a result, it is possible to be executed in an ambiently powered WSN. However, this comes at the cost of increased delay, since in order to get a good estimate of the average consensus fixed point, the algorithm must be repeated multiple times.

One interesting future direction of this work is to study the convergence properties of the aforementioned framework more thoroughly. In particular, since -as we saw- the convergence of the algorithms is in good part dependent on how the elements of the vector are assigned to each WSN node and on the statistics of matrix $\mathbf{P}$, an interesting question that arises is how to *optimaly* balance the computations in order to achieve faster and more robust convergence. In addition, this work would be enhanced by studying a more general fixed point problem than the one of Eq. 6.1, namely

$$\mathbf{x}^{(l)} = f(\mathbf{x}^{(l\ \ 1)}), \tag{6.2}$$

where $f : X \;\; ! \;\; X$ a mapping. Considering what was discussed in this work, the asynchronous variant of Eq. 6.2 would be

$$\mathbf{x}^{(l)} = \mathbf{\Psi}^{(l)} f(\mathbf{x}^{(l \;\; 1)}) + (\mathbf{I} \quad \mathbf{\Psi}^{(l)}) \mathbf{x}^{(l \;\; 1)}. \tag{6.3}$$

Hence, it would be of great interest to analyze and study the conditions under which Eq. 6.3 converges.

Bibliography

[1] G. Vannucci, A. Bletsas, and D. Leigh, "A software-defined radio system for backscatter sensor networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2170–2179, Jun. 2008.

[2] C. Konstantopoulos, E. Kampianakis, E. Koutroulis, and A. Bletsas, "Wireless sensor node for backscattering electrical signals generated by plants," in *Proc. IEEE Sensors Conf. (Sensors)*, Baltimore, MD, USA, Nov. 2013.

[3] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," in *Proc. ACM SIGCOMM*, Hong Kong, China, 2013, pp. 39–50.

[4] J. Kimionis, A. Bletsas, and J. N. Sahalos, "Increased range bistatic scatter radio," *IEEE Trans. Commun.*, vol. 62, no. 3, pp. 1091–1104, Mar. 2014.

[5] E. Kampianakis, J. Kimionis, K. Tountas, C. Konstantopoulos, E. Koutroulis, and A. Bletsas, "Wireless environmental sensor networking with analog scatter radio & timer principles," *IEEE Sensors J.*, vol. 14, no. 10, pp. 3365—3376, Oct. 2014.

[6] S. N. Daskalakis, S. D. Assimonis, E. Kampianakis, and A. Bletsas, "Soil moisture wireless sensing with analog scatter radio, low power, ultra-low cost and extended communication ranges," in *Proc. IEEE Sensors Conf. (Sensors)*, Valencia, Spain, Nov. 2014, pp. 122–125.

[7] N. Fasarakis-Hilliard, P. N. Alevizos, and A. Bletsas, "Coherent detection and channel coding for bistatic scatter radio sensor networking," in *Proc. IEEE Int. Conf. Communications*, Jun. 2015, pp. 4895–4900.

[8] S. N. Daskalakis, S. D. Assimonis, E. Kampianakis, and A. Bletsas, "Soil moisture scatter radio networking with low power," *IEEE Trans. Microwave Theory Tech.*, vol. 64, no. 7, pp. 2338–2346, Jul. 2016.

[9] V. Papageorgiou, A. Nichoritis, P. Vasilakopoulos, G. Vougioukas, and A. Bletsas, "Towards ambiently powered inference on wireless sensor networks: Asynchrony is the key!" in *5th IEEE International Workshop on Wireless Communications and Networking in Extreme Environments (WCNEE)*, Jul. 2021.

[10] J. S. Yedidia, "Message-passing algorithms for inference and optimization," *Journal of Statistical Physics*, vol. 145, no. 4, pp. 860–890, Nov. 2011.

[11] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theor.*, vol. 47, pp. 498–519, Feb. 2001.

[12] D. Bickson, "Gaussian belief propagation: theory and application," Ph.D. dissertation, Hebrew University of Jerusalem, Oct. 2008.

[13] B. Li and Y.-C. Wu, "Convergence analysis of gaussian belief propagation under high-order factorization and asynchronous scheduling," *IEEE Trans. Signal Processing*, vol. 67, no. 11, pp. 2884–2897, Jun. 2019.

[14] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[15] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[16] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[18] B. J. Frey, F. R. Kschischang, H. andrea Loeliger, and N. Wiberg, "Factor graphs and algorithms," in *Proc. 35th Allerton Conf. on Communications, Control, and Computing, (Allerton*, 1998, pp. 666–680.

[19] R. G. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Massachusetts Institute of Technology, 1963.

[20] J. Du, S. Ma, Y.-C. Wu, S. Kar, and J. M. F. Moura, "Convergence analysis of distributed inference with vector-valued gaussian belief propagation," *Journal of Machine Learning Research*, vol. 18, no. 172, pp. 1–38, Apr. 2018.

[21] B. C. Levy, *Principles of Signal Detection and Parameter Estimation*. New York: Springer, 2008.

[22] U. A. Khan, "High-dimensional consensus in large-scale networks: Theory and applications," Ph.D. dissertation, ECE Department, Carnegie Mellon University, Aug. 2009, advisor: J. M. Moura.

[23] S. Kar and J. M. Moura, "Consensus + innovations distributed inference over networks: cooperation and sensing in networked systems," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 99–109, Apr. 2013.

[24] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[25] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, Jan. 2007.

[26] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. USA: Society for Industrial and Applied Mathematics, 2000.

[27] Z. Chen, L. Cai, and C. Zhao, "Convergence time of average consensus with heterogeneous random link failures," *Automatica*, vol. 127, p. 109496, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109821000169

[28] 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.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.