Methods for finite-time average consensus protocols design, network robustness assessment and network topology reconstruction

Thi-Minh-Dung Tran

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Automatique-Productique**

Arrêté ministériel : 7 août 2006

Présentée par

TRAN Thi-Minh-Dung

Thèse dirigée par **M. Carlos CANUDAS DE WIT**
et codirigée par **M. Alain Y. KIBANGOU**

préparée au sein **GIPSA-Lab, Département Automatique**
et de **Électronique, Électrotechnique, Automatique, Traitement du Signal**

Methods for Finite-time Average Consensus Protocols Design, Network Robustness Assessment and Network Topology Reconstruction

Thèse soutenue publiquement le **26 Mars 2015**,
devant le jury composé de :

M. Christian COMMAULT
Professeur, GIPSA-Lab, Université Grenoble Alpes (Grenoble, France), Président

M. Jean-Marie GORCE
Professeur, INSA-Lyon, (Lyon, France), Rapporteur

M. Alessandro GIUA
Professeur, Aix-Marseille University (France) - University of Cagliari (Italy), Rapporteur

M. Antoine GIRARD
Maître de conférences, HDR, Université Grenoble Alpes (Grenoble, France), Examinateur

M. Walid HACHEM
Directeur de recherche CNRS Telecom Paris Tech (Paris, France), Examinateur

M. Alain KIBANGOU
Maître de Conférences, Université Grenoble Alpes (Grenoble, France), Encadrant

M. Fabio MORBIDI
Maître de conférences, Université de Picardie Jules Verne (France), Examinateur

Table of contents

List of Acronyms

MAS Multi-agent System

FC Fusion Center

WSN Wireless Sensor Network

ADMM Alternating Direction of Multipliers Method

MSE Mean Square Error

FFT Fast Fourier Transform

SRG Strongly Regular Graph

SREL Success rate of existing links

SRNL Success rate of non-existing links

Acknownledgement

First and foremost, I would like to express my appreciation to my supervisor, Dr. Alain Y. Kibangou, for his guidance, advice, kindness and support during my study in Necs-team, Gipsa-Lab of the University of Grenoble. When I arrived here, I was very short on this research domain. But, Alain guided me how to get insight into this problem step by step without hesitation. He taught me how to do a research. All of my works in this dissertation cannot be accomplished without his support.

I would like to express my gratitude to Prof. Caslos Canuda-de-Wits for his help, guidance and the opportunities he gave me to improve myself during the time in Necs-team.

I would like to thank my friends, especially all members in Necs-team for their encouragement, discussion and friendship.

My family has been a constant source of inspiration and support for me through the good and hard times. I owe a deep debt for their love. Without them, I cannot do anything, including this thesis.

Most of all, I would like to express my deepest gratitude to my husband AN. He gives me so much more than I ever expected. Thanks for his understanding, support and staying by my side up till now.

Abstract

Consensus of Multi-agent System (MAS)s has received tremendous attention during the last decade. Consensus is a cooperative process in which agents interact in order to reach an agreement. Most of studies are committed to the analysis of the steady-state behavior of this process. However, during the transient of this process a huge amount of data is produced. In this thesis, our aim is to exploit data produced during the transient of asymptotic average consensus algorithms in order to design finite-time average consensus protocols, assess the robustness of the graph, and eventually recover the topology of the graph in a distributed way.

Finite-time average consensus guarantees a minimal execution time that can ensure the efficiency and the accuracy of sophisticated distributed algorithms in which it is involved. We first focus on the configuration step devoted to the design of consensus protocols that guarantee convergence to the exact average in a given number of steps. By considering networks of agents modelled with connected undirected graphs, we formulate the problem as the factorization of the averaging matrix $\mathbf{J}_N = \frac{1}{N}\mathbf{1}\mathbf{1}^T$ and investigate distributed solutions to this problem. Since communicating devices have to learn their environment before establishing communication links, we suggest the usage of learning sequences in order to solve the factorization problem. Then a gradient back-propagation-like algorithm is proposed to solve a non-convex constrained optimization problem. We show that any local minimum of the cost function provides an accurate factorization of the averaging matrix.

By constraining the factor matrices to be Laplacian-based consensus matrices, it is now well known that the factorization of the averaging matrix is fully characterized by the nonzero Laplacian eigenvalues. Therefore, solving the factorization of the averaging matrix in a distributed way with such Laplacian matrix constraint allows estimating the spectrum of the Laplacian matrix. Since that spectrum can be used to compute some robustness indices (number of spanning trees and effective graph resistance also known as Kirchoff index), the second part of this dissertation is dedicated to network robustness assessment through distributed estimation of the Laplacian spectrum. The problem is posed as a constrained consensus problem formulated in two ways. The first formulation (direct approach) yields a non-convex optimization problem solved in a distributed way

by means of the method of Lagrange multipliers. The second formulation (indirect approach) is obtained after an adequate re-parametrization. The problem is then convex and solved by using the distributed sub-gradient algorithm and the Alternating Direction of Multipliers Method (ADMM). Furthermore, three cases are considered: the final average value is perfectly known, approximated, or completely unknown. We also provide a way for computing the multiplicities of the estimated eigenvalues by means of integer programming.

In this spectral approach, given the Laplacian spectrum, the network topology can be reconstructed through estimation of Laplacian eigenvector. In particular, we study the reconstruction of the network topology in the presence of anonymous nodes.

The efficiency of the proposed solutions is evaluated by means of simulations. However, in several cases, convergence of the proposed algorithms is slow and needs to be improved in future works. In addition, the indirect approach is not scalable to very large graphs since it involves the computation of roots of a polynomial with degree equal to the size of the network. However, instead of estimating all the spectrum, it can be possible to recover only a few number of eigenvalues and then deduce some significant bounds on robustness indices.

Résumé

Durant la dernière décennie, une attention considérable a été consacrée au consensus des systèmes multi-agents. Le consensus est un processus coopératif dans lequel les agents interagissent afin de parvenir à un accord. La plupart des études ont été orientées vers l'analyse en régime permanent de ce processus d'agrément. Or, durant le régime transitoire, les algorithmes de consensus à convergence asymptotique génèrent un grand nombre de données. Dans cette thèse, notre objectif est d'exploiter ces données afin de concevoir des protocoles de consensus moyenne en temps fini, évaluer la robustesse du graphique, et éventuellement reconstituer la topologie du réseau de manière distribuée.

Le consensus de moyenne en temps fini garantit un temps d'exécution minimal qui peut assurer l'efficacité et la précision des algorithmes distribués complexes dans lesquels il est utilisé comme une sous-composante. Nous nous concentrons d'abord sur l'étape de configuration consacrée à la conception de protocoles de consensus qui garantissent la convergence vers la moyenne de la condition initiale dans un nombre donné d'étapes. En considérant des réseaux d'agents modélisés avec des graphes non orientés connectés, nous formulons ce problème comme étant un problème de factorisation de la matrice de moyenne $\mathbf{J}_N = \frac{1}{N}\mathbf{1}\mathbf{1}^T$ et étudions des solutions distribuées à ce problème. Puisque, les objets communicants doivent apprendre leur environnement avant d'établir des liens de communication, nous suggérons l'utilisation de séquences d'apprentissage afin de résoudre le problème de la factorisation. Ensuite, un algorithme semblable à l'algorithme de rétro-propagation du gradient est proposé pour résoudre un problème d'optimisation non convexe sous contrainte. Nous montrons que tout minimum local de la fonction de coût donne une factorisation exacte de la matrice de moyenne.

En contraignant les matrices facteur à être comme les matrices de consensus basées sur la matrice laplacienne, il est maintenant bien connu que la factorisation de la matrice de moyenne est entièrement caractérisée par les valeurs propres non nulles de la matrice Laplacienne du graphe. Par conséquent, la résolution de la factorisation de la matrice de moyenne de manière distribuée, avec une telle contrainte, permet d'estimer le spectre de la matrice Laplacienne. Puisque, spectre peut être utilisé pour calculer des indices de robustesse (Nombre d'arbres couvrant et résistance effective du graphe, aussi connu comme l'indice de Kirchhoff), la deuxième partie de cette thèse est consacrée à l'évaluation de

la robustesse du réseau à travers l'estimation distribuée du spectre du Laplacien. Le problème est posé comme un problème de consensus sous contrainte formulé de deux façons différentes. La première formulation (approche directe) conduit à un problème d'optimisation non-convexe résolu de manière distribuée au moyen de la méthode des multiplicateurs de Lagrange. La seconde formulation (approche indirecte) est obtenue après une reparamétrisation adaptée. Le problème devient alors convexe et est résolu en utilisant l'algorithme du sous-gradient distribué et la méthode de direction alternée de multiplicateurs (ADMM). En outre, trois cas sont considérés: la valeur moyenne est parfaitement connue, elle est approximativement connue, ou complètement inconnue. Nous fournissons également une technique de calcul des multiplicités des valeurs propres estimées au moyen d'une programmation linéaire en nombres entiers.

Dans cette approche spectrale, compte tenu du spectre du Laplacien, la topologie du réseau peut être reconstruite à travers l'estimation des vecteurs propres du Laplacien. Nous proposons une nouvelle approche permettant à chaque nœud du réseau de pouvoir reconstruire la topologie du réseau même en cas de présence de nœuds anonymes.

L'efficacité des solutions proposées est évaluée au moyen de simulations. Cependant, dans plusieurs cas, la convergence des algorithmes proposés est lente et doit être améliorée dans les travaux futurs. La passage à l'échelle des méthodes développées est une question crucial qui mériterait d'être approfondie. Cependant, au lieu d'estimer tout le spectre, il peut être plus judicieux de ne restreindre l'estimation qu'à quelques valeurs propres et de pouvoir en déduire des bornes significatives sur les mesures de robustesse.

List of Notations

$\triangleq$	Definition
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{++}$	Set of positive real numbers
$\mathbf{x}$	Vector $\mathbf{x}$
x_i	The i^{th} element of the vector $\mathbf{x}$
$\mathbf{X}$	Matrix $\mathbf{X}$
$\mathbf{X}^T$	Transpose of the matrix $\mathbf{X}$
$\|\mathbf{x}\|$	Euclidien norm, $\|\mathbf{x}\| \triangleq \sqrt{\mathbf{x}^T\mathbf{x}}$
G	The (undirected) graph $G(V, E)$
V	Set of vertices (or nodes or agents)
E	Set of edges (or links)
d_i	Degree of node i
d_{max}	Maximum degree, $d_{max} = \max_i d_i$
N_i	Neighborhood of node i
N	Number of nodes
$d(G)$	Diameter of graph G
$\|\mathbf{S}\|$	The number of elements in the set $\mathbf{S}$ (i.e. cardinality of the set $\mathbf{S}$)
$\mathbf{A}$	Adjacency matrix
$\mathcal{D}$	Degree matrix
$\mathbf{L}$	Laplacian matrix
$sp(\mathbf{L})$	Laplacian Spectrum
Λ	The set of nonzero distinct Laplacian eigenvalues
$\lambda_i(\mathbf{L})$	The i^{th} eigenvalue of the Laplacian matrix
$\mathbf{W}$	Consensus Matrix
w_{ij}	The element on row i and column j of the matrix $\mathbf{W}$ (the (i, j) element of matrix $\mathbf{W}$)
$sp(\mathbf{W})$	The spectrum of matrix $\mathbf{W}$

$\mathcal{S}_G$	The set of matrices, $\mathcal{S}_G = \{\mathbf{W} \in I\!R^{N\times N} \vert w_{ij} = 0 \text{ if } (i,j) \notin E \text{ and } i \neq j\}$
$\mathbf{1}_N$	Column vector with all N elements equal to 1
$\rho(\mathbf{A})$	Spectral radius $\rho(\mathbf{A}) = \max\{\vert\lambda_1(\mathbf{A})\vert, \ldots, \vert\lambda_N(\mathbf{A})\vert\}$
$\mathbf{J}_N$	Averaging matrix, $\mathbf{J}_N = \frac{\mathbf{1}\mathbf{1}^T}{N}$
$\mathbf{I}$	Identity matrix (the $N \times N$ matrix)
$\mathbf{I}_N$	Identity matrix in $I\!R^{N\times N}$
$\mathcal{K}_v(\mathcal{K}_e)$	Vertex (Egde) Connectivity
ξ	Number of spanning trees
$\mathcal{R}$	Effective graph resistance
$\Omega_C[\mathbf{x}]$	Euclidian projection of $\mathbf{x}$ on the contraint set $\mathbf{C}$, $\Omega_C[\mathbf{x}] = \arg\min_{\mathbf{z}\in\mathbf{C}} \Vert\mathbf{x} - \mathbf{z}\Vert^2 \vert$
$\odot$	Khatri-Rao product
$\otimes$	Kronecker product
$\circ$	Hamadard product
$vec(.)$	operator stacks the column of its matrix argument
$vecd(.)$	the column vector built with the diagonal entries of the matrix in the argument.

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Context of the thesis

1.1.1 Multi-agent systems

Multi-agent systems (MASs) have received a growing interest in the last decades. They are developed for the demand of flexibility, robustness, and re-configuration features that appear in various applications domains including manufacturing, logistics, smart power grids, building automation, disaster relief, intelligent transportation systems, surveillance, environmental monitoring and exploration, infrastructure security and protection, etc.. A MAS is a system composed of multiple interacting intelligent agents (sensors, plants, vehicles, robots, etc.) and their environment as shown in Figure 1.1.

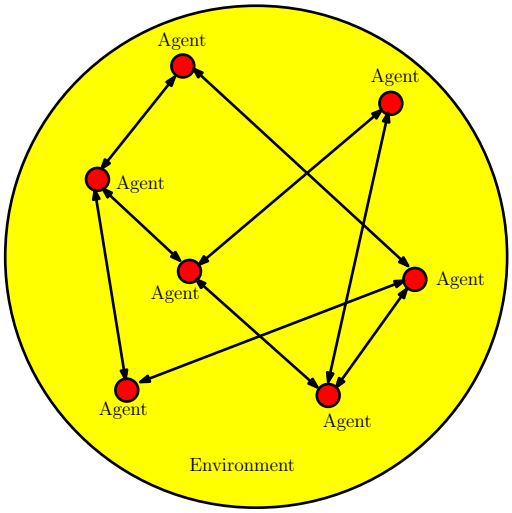


Figure 1.1: General architecture of a MAS.

An intelligent agent possesses the following key characteristics [26]:

- *Autonomy*: the ability to partially operate without the intervention of a human.
- *Reactivity*: the ability to react to changes in its environment and to behave correctly in order to satisfy its goal.
- *Pro-activeness*: the ability to take a responsibility for setting its own goals.
- *Social ability*: the ability to interact with humans and other agents.
- *Local views*: No agent has a full global view of the system, or the system is too complex for an agent to make practical use of such a knowledge.

A MAS can deal with tasks that are difficult or even impossible to be accomplished by an individual agent. During the recent decades, MASs have gained a widespread interest in

many disciplines such as mathematics, physics, biology, computer science, social science. An increasing range of research topics in MASs includes cooperation and coordination, distributed computation, automatic control, wireless communication networks, etc..

In a typical centralized structure, a Fusion Center (FC) collects all measurements from the agents and then makes the final computations. However, due to the high information flow to FC, congestion can arise. Such a structure is vulnerable to FC failure. Also, the hardware requirements to build wireless communications can be one of reasons for an increase in the cost of the devices and thus, a higher overall cost of the network. For these reasons, a centralized structure can be inefficient. Hence, the research trend of MASs have shifted to *decentralized structure* where the interaction between agents is implemented locally without global knowledge. A good example is Wireless Sensor Network (WSN)s, which find broad application domains such as military applications (battlefield surveillance, monitoring friendly forces, equipment and ammunition,etc.), environment applications (forest fire detection, food detection,etc.), health applications (telemonitoring of human physiological data,etc.), home automation, formation control, etc.. Figure 1.2 depicts a WSN that collects data for the air quality, light intensity, sound volume, heat, precipitation and wind.

Figure 1.2: A network of wireless sensors on the light poles all over the city of Chicago. (Source:http://mostepicstuff.com/chicago-city-installing-smart-sensors-city-wide-to-monitor-everything)

1.1.2 Motivation

There are three points that attract a great attention of the research community:

(A) **Based on local information and interactions between agents, how can all agents reach an agreement?**

This problem is called *consensus problem*, which is to design a network protocol based on the local information obtained by each agent such that all agents finally reach an agreement on certain quantities of interest.

Consensus problems of MASs have received tremendous attention from various research communities due to their broad applications in many areas including multi-sensors data fusion [64], flocking behavior of swarms [6, 38, 59, 60], multi-vehicle formation control [17, 54, 62], distributed computation [5, 10, 48], rendez-vous problems [15] and so on. More specifically, average consensus algorithms (i.e. the consensus value corresponds to the average of the initial states) are commonly used as building block for distributed control, estimation or inference algorithms.

In the recent literature, one can find average consensus algorithms embedded in the distributed Kalman filter, [61]; Distributed Least Squares algorithms, [9]; Distributed Alternating Least Squares for tensors factorization, [45]; Distributed Principal Component Analysis, [49]; or distributed joint input and state estimation, [24] to cite few. However, the asymptotic convergence of the consensus algorithms is not suitable for these kinds of sophisticated distributed algorithms. A slow asymptotic convergence can not ensure the efficiency and the accuracy of the algorithms, which can lead to other unexpected effects. For example, regarding to the WSNs, a reduction in the total number of iterations until convergence can lead to a reduction in the total amount of energy consumption of the network, which is essential to guarantee a longer life time for the entire network. On the other hand, the protocols that guarantee a minimal execution time are much more appealing than those ensuring asymptotic convergence. For this purpose, several contributions dedicated to **finite-time consensus** have been recently published in the literature, meaning that, consensus is obtained in a finite number of iterations.

(B) **Can we assess the robustness of a network?**

Generally, we are surrounded by networks of different kinds (social networks, sensor networks, power networks, etc.). Therefore, the most important thing is that these networks are robust. Meaning that, they can undergo through damage or failure. Hence, the terminology *robustness* rapidly invades the research community.

Definition 1

> *Robustness is the ability of a network to continue performing well when it is subject to failures or attacks [21].*

In order to assess whether the network is robust, measuring the network robustness is needed [21]. There are various proposed approaches to measure network robustness metrics. However, the most popular approach in the literature is based on the analysis of the graph. **From data produced during consensus protocol can we infer network robustness?**

(C) **Can we estimate the network topology?**

Network topology is usually a schematic description of the arrangement of a network, including its nodes and connecting edges. Network topology may describe how data is transferred between these nodes.

Network topology identification refers to detecting and identifying the interested network elements and the relationship between elements in this network and represents the topology construction in an appropriate form. Therefore, identification of networks of systems becomes a growing attractive task for solving many problems in different science and engineering domains.

Theoretically, there are some matrices that are associated with the graph representing a network. Therefore, one can find a method to identify the network topology via the state matrix in a state space representation of linear dynamic systems, which is related to the network topology. Eigenvalue decomposition of a matrix shows that the structure of this matrix is linked to both eigenvalues and eigenvectors of that matrix. In the other words, estimating both the eigenvalues and the eigenvectors of network matrix can obviously infer the network structure.

1.1.3 Goals of the thesis

It is well-known that consensus protocols being iterative, a huge amount of data is produced. However, most of studies focus only on convergence properties of such algorithms. With data collected from each iteration of the asymptotic consensus protocol in a distributed way, we aim to:

- Design finite-time average consensus protocols.
- Assess the network robustness.
- Reconstruct the network topology.

1.2 The consensus problem

Consensus issue in networks of autonomous agents has been widely investigated in various fields, including computer science and engineering. In such networks, according to an a priori specified rule, also called protocol, each agent updates its state based on the information received from its neighbors with the aim of reaching an agreement to a common value. When the common value corresponds to the average of the initial states, average consensus is to be achieved.

1.2.1 Preliminaries

Let us consider a network modelled as a graph. In what follows, we first give some basic notations and definitions from graph theory [32, 52].

Given a connected undirected graph $G(V, E)$, where $V = \{1, 2, \ldots, N\}$ is the set of vertices (agents or nodes), and $E \subseteq V \times V$ is the set of edges (links) between agents. The set of neighbors of node i and its degree are denoted by $N_i = \{j | (i, j) \in E\}$ and $d_i = |N_i|$ respectively.

It is common to resort to some matrices for characterizing graphs. That is the case of adjacency matrix $\mathbf{A}$, degree matrix $\mathcal{D}$ of the graph, which has vertex degrees $d_i, i \in V$ on its diagonal and zeroes elsewhere, Laplacian Matrix $\mathbf{L} = \mathcal{D} - \mathbf{A}$, clearly defined in Chapter 2.

Through out this thesis, an undirected graph $G(V, E)$ is mainly considered.

Example:

Consider an arbitrary network of 5 agents communicating with each other as described in Figure 1.3. Each agent has an initial value. A consensus protocol is an interaction rule that specifies the information exchange between an agent and all of its neighbors on the network to reach an agreement regarding a certain quantity of interest that depends on the state of all agents. Informally, despite the initial values of all agents, they converge to the common value (in this case, the average of initial values).

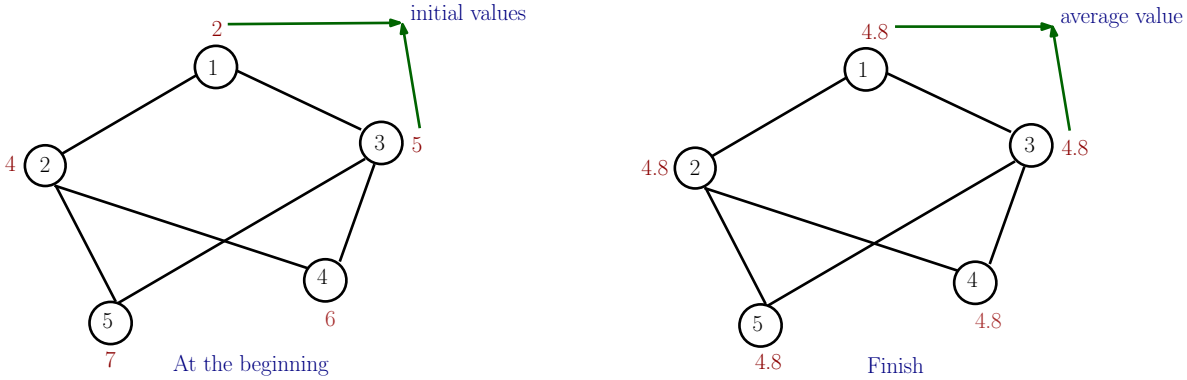


Figure 1.3: Average consensus in a network: initial condition (left) and steady state (right).

1.2.2 Consensus algorithms

The literature on consensus protocols can be organized as in Figure 1.4.

In this thesis, we concentrate on the consensus problem for discrete-time systems with fixed topology.

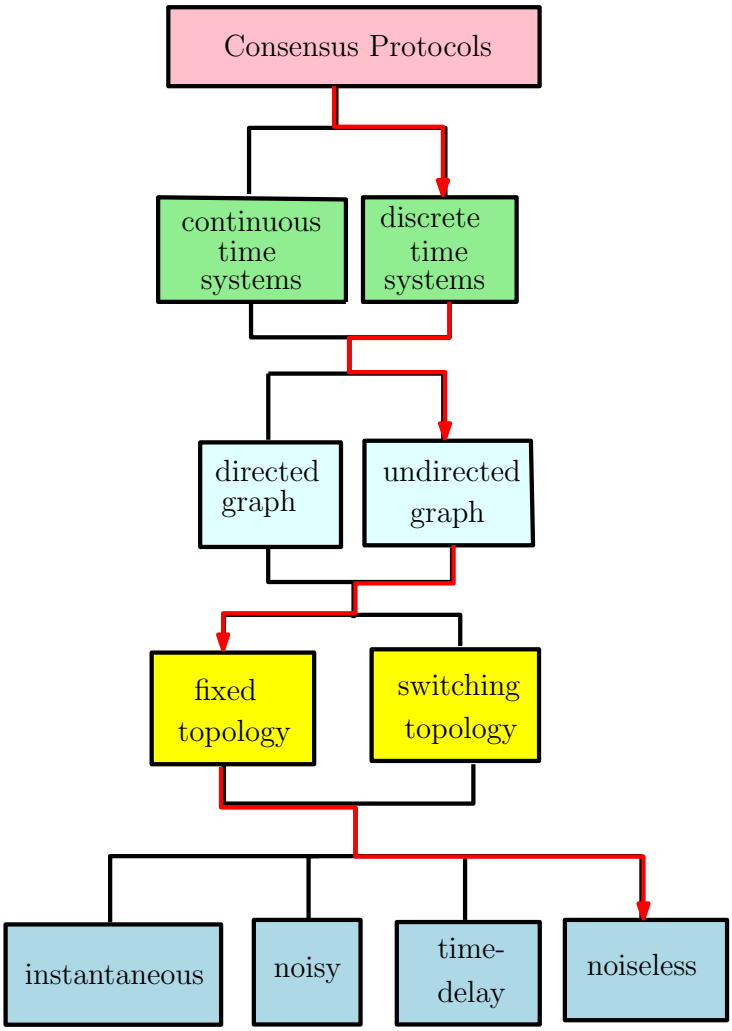


Figure 1.4: Classification of Consensus protocols.

Given a graph $G(V, E)$, each node has an associated value x_i defined as the state of node i. Let $\mathbf{x}(0) = [x_1(0), x_2(0), \ldots, x_N(0)]^T$ be the vector of initial states of the given network. In general, given the initial values at each node $x_i(0), i \in V$, the main task is to compute the final consensus value using *distributed linear iterations*. Each iteration involves local communication between nodes. In particular, each node repeatedly updates its value as a linear combination of its own value and those of its neighbors. The main benefit of using a linear-iteration scheme is that, at each time-step, each node only has to transmit a single value to each of its neighbors. The linear iteration-based consensus update equation is expressed as:

$$x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), \qquad i = 1, 2, \ldots, N$$

or equivalently in matrix form:

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k),$$

where: $\mathbf{W}(k)$ is the matrix with entries $w_{ij}(k) = 0$ if $(i,j) \notin E$ and $\sum_{j \in N_i \cup \{i\}} w_{ij}(k) = 1$.

The asymptotic consensus is reached if $\lim_{k\to\infty} \mathbf{x}(k) = \mu\mathbf{1}$, meaning that all the nodes agreed on the value μ. When μ is equal to the average of the initial values, i.e.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i(0),$$

it is called average consensus problem.

The distributed average consensus algorithm is described as follows:

Algorithm 1 (*Distributed average consensus algorithm*)

1. ***Input***:
   - *Network topology modelled with a graph* $G(V,E)$,
   - *Initial values* $\mathbf{x}(0)$,
   - *Maximum number of iterations* k_{max}.
2. ***Initialization***: *Define weight consensus matrix* $\boldsymbol{W}(k), k = 0, 1, \ldots$.
3. ***Set*** $k = 0$.
4. ***While*** $k < k_{max}$ ***do***
   - *Update:* $x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), i = 1, \ldots, N.$
   - $k = k + 1$;
5. ***end***
6. ***Output***: $\mathbf{x}(k_{max}) = \mu\boldsymbol{1}$.

1.2.3 Convergence conditions

According to the classification of consensus algorithms in Figure 1.4, the convergence analysis is also studied for specified categories such as static topology and dynamic topology, directed and undirected topologies. In this thesis, we focus on distributed discrete-time average consensus protocols for a network modelled by an undirected graph with fixed topologies. Meaning that, the evolution of the $\mathbf{x}(k)$ can be written in matrix form as follows:

$$\mathbf{x}(k+1) = \mathbf{W}\mathbf{x}(k), \quad k = 0, 1, \ldots \tag{1.2.1}$$

or equivalently,

$$\mathbf{x}(k) = \mathbf{W}^k \mathbf{x}(0). \tag{1.2.2}$$

Generally speaking, the system is said to achieve distributed consensus asymptotically if

$$\lim_{k\to\infty} \mathbf{x}(k) = \lim_{k\to\infty} \mathbf{W}^k \mathbf{x}(0) = \mathbf{1}\boldsymbol{c}^T \mathbf{x}(0), \tag{1.2.3}$$

where $\mathbf{1} \in R^{N\times 1}$ is the all-ones column vector and $\boldsymbol{c}^T$ is some constant row vector. Therefore, it is good to view the conditions on the weight matrix $\mathbf{W}$ ensuring the convergence of the consensus protocol as well as the average consensus protocol:

$$\lim_{k\to\infty} \mathbf{W}^k = \mathbf{1}\boldsymbol{c}^T. \tag{1.2.4}$$

The convergence conditions are described as follows:

Theorem 1.2.1 (*[74]*)

Consider the linear iteration protocol (1.2.1), distributed consensus is achieved if and only if the weight consensus matrix $\boldsymbol{W}$ satisfies the following conditions:

1. $\boldsymbol{W1} = \mathbf{1}$
2. $\rho(\boldsymbol{W} - \mathbf{1}\boldsymbol{c}^T) < 1$,

where $\boldsymbol{c}$ is chosen so that $\mathbf{1}^T\boldsymbol{c} = 1$ and $\rho(\boldsymbol{W} - \mathbf{1}\boldsymbol{c}^T)$ is the spectral radius of $\boldsymbol{W} - \mathbf{1}\boldsymbol{c}^T$.

To sum up, the weight matrix has row-sum equal to 1, 1 is a simple eigenvalue of $\mathbf{W}$ and all other eigenvalues are strictly less than one in magnitude. It means that the weight matrix $\mathbf{W}$ is a row-stochastic matrix.

When $\boldsymbol{c} = \frac{1}{N}\mathbf{1}$, the system is said to achieve average consensus (i.e the consensus value is the average of all initial values):

$$\lim_{k\to\infty} \mathbf{W}^k = \frac{1}{N}\mathbf{1}\mathbf{1}^T. \tag{1.2.5}$$

In order to achieve average consensus, beside the conditions in Theorem 1.2.1, the left and right eigenvector of $\mathbf{W}$ corresponding to eigenvalue 1 are $\mathbf{c}^T$ and $\mathbf{1}$, respectively.

Theorem 1.2.2

The equation (1.2.5) holds if and only if [74]:

1. $\boldsymbol{1}^T\boldsymbol{W} = \mathbf{1}$
2. $\boldsymbol{W1} = \mathbf{1}$
3. $\rho(\boldsymbol{W} - \frac{1}{N}\mathbf{1}\mathbf{1}^T) < 1$

A necessary and sufficient condition to ensure the convergence of the average consensus protocol for undirected graphs is that the weight consensus matrix $\mathbf{W}$ is a doubly stochastic matrix.

1.2.4 Design of the weight matrix.

In the literature, there are some works devoted to the design of the weight matrix $\mathbf{W}$ that satisfies the convergence conditions of the average consensus algorithms. For instance, in [74, 75], it has been shown that the following weight matrices satisfy the convergence conditions pointed out above.

1. Maximum-degree weights:

   An approach to design the weight matrix $\mathbf{W}$ in a graph with fixed topology (time-invariant topology) consists in assigning a weight on each edge equal to the inverse of the maximum degree of the network, i.e.,

   $$w_{ij} = \begin{cases} \frac{1}{d_{max}+1} & \text{if} \quad j \in N_i \\ 1 - \frac{d_i}{d_{max}+1} & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \tag{1.2.6}$$

   where $d_{max} = \max_{i \in V} d_i \leq N$. We can see that before running the consensus algorithm, the maximum degree d_{max} should be known *in a priori*. The required information on the graph topology is global, but can be obtained in a distributed way by running a max consensus protocol [63].

2. Metropolis hasting weights:

   The metropolis weights for a graph with a time-invariant topology are proposed in [74]:

   $$w_{ij} = \begin{cases} \frac{1}{1+\max\{d_i,d_j\}}, & \text{if} \quad j \in N_i \\ 1 - \sum_{j \in N_i} \frac{1}{1+\max\{d_i,d_j\}}, & \text{if} \quad i = j \\ 0, & \text{otherwise} \end{cases} \tag{1.2.7}$$

   Here, the nodes do not need any global information of the communication graph, not even the number of the vertices N. They only need to know their own degree and that of their neighbors to compute the weights. The metropolis weights are very simple to compute and suited to a distributed implementation.

3. Constant edge weights:

By a result in [74], the simplest approach is to set all the edge weights for neighboring nodes equal to a constant α. This is the most widely applied model for the weight matrix in both time-variant and time-invariant topologies. The weight matrix is defined as follows:

$$w_{ij} = \begin{cases} \alpha, & \text{if} \quad j \in N_i \\ 1 - \alpha|N_i|, & \text{if} \quad i = j \\ 0, & \text{otherwise} \end{cases} \tag{1.2.8}$$

where $|\cdot|$ denotes the cardinality of the set. The weight matrix can be expressed in matrix form as follows:

$$\mathbf{W} = \mathbf{I}_N - \alpha\mathbf{L}, \tag{1.2.9}$$

where $\mathbf{L}$, $\mathbf{I}_N$ are the Laplacian matrix of the graph G and the $N \times N$ identity matrix respectively. In general, $\alpha > 0$ for the convergence condition 1.2.2 to hold. Additionally, we can express the eigenvalues of $\mathbf{W}$ in terms of those of $\mathbf{L}$:

$$\lambda_i(\mathbf{W}) = 1 - \alpha\lambda_i(\mathbf{L}), i = 1, \ldots, N,$$

where $\lambda_i(.)$ denotes the i-th largest eigenvalue of $\mathbf{L}$. Therefore,

$$\begin{aligned} \rho(\mathbf{W} - \frac{1}{N}\mathbf{1}\mathbf{1}^T) &= \max\{\lambda_2(\mathbf{W}), -\lambda_N(\mathbf{W})\} \\ &= \max\{1 - \alpha\lambda_2(\mathbf{L}), \alpha\lambda_N(\mathbf{L}) - 1\}. \end{aligned} \tag{1.2.10}$$

The value of α that can ensure the convergence condition of the consensus protocols is $0 < \alpha < \frac{2}{\lambda_N(\mathbf{L})}$. There are some simple choices that do not require the knowledge of the Laplacian spectrum, for example, the upper bound of the largest Laplacian eigenvalue $\lambda_N \leq 2d_{max}$. Hence, we can have a simple choice of α as $0 < \alpha < \frac{1}{d_{max}}$. Moreover, following the result in [74], the best possible constant edge weight in terms of convergence speed is given by the following α:

$$\alpha = \frac{2}{\lambda_N(\mathbf{L}) + \lambda_2(\mathbf{L})}. \tag{1.2.11}$$

In the analysis of consensus problems, convergence rate is an important index to evaluate the performance of a consensus protocol. Therefore, there are several works dealing with accelerating the rate of convergence of the consensus protocol by solving some optimization problems in a centralized way or using Chebyshev polynomials.

Since the conditions of the weight consensus matrix $\mathbf{W}$ obviously influence on the convergence of consensus algorithms, the main direction of research is now shifted to the computation of $\mathbf{W}$ in order to improve the convergence rate, [47, 53, 74].

A. [74] proposed an optimization method to obtain the optimum weight matrix $\mathbf{W}$ achieving average consensus in linear time-invariant topologies as the solution of a semi-definite convex programming.

$$\begin{array}{ll} \min\limits_{\mathbf{W}} & \rho(\mathbf{W} - \frac{1}{N}\mathbf{1}\mathbf{1}^T) \\ \text{subject to} & \mathbf{W} \in \mathcal{S}_G, \quad \mathbf{1}^T\mathbf{W} = \mathbf{1}^T, \quad \mathbf{W}\mathbf{1} = \mathbf{1}. \end{array}$$

The condition $\mathbf{W} \in \mathcal{S}_G$ expresses the constraint on the sparsity pattern of the matrix $\mathbf{W}$ with the set $\mathcal{S}_G$ defined as follows:

$\mathcal{S}_G = \{\mathbf{W} \in R^{N\times N} | w_{ij} = 0 \text{ if } (i,j) \notin E \text{ and } i \neq j\}$.

B. In [47], the author proposed an interesting method to accelerate the speed of convergence through polynomial filtering. The main idea is to apply a polynomial filter on the consensus matrix $\mathbf{W}$ that will shape its spectrum in order to increase the convergence rate.

Denote by $p_k(\lambda(\mathbf{W}))$ the polynomial filter of degree k that is applied on the spectrum of $\mathbf{W}$,

$$p_k(\lambda(\mathbf{W})) = \alpha_0 + \alpha_1\lambda(\mathbf{W}) + \alpha_2\lambda^2(\mathbf{W}) + \ldots + \alpha_k\lambda^k(\mathbf{W}). \tag{1.2.12}$$

The matrix polynomial is defined as:

$$p_k(\mathbf{W}) = \alpha_0\mathbf{I}_N + \alpha_1\mathbf{W} + \alpha_2\mathbf{W}^2 + \ldots + \alpha_k\mathbf{W}^k. \tag{1.2.13}$$

where $\alpha_0, \ldots, \alpha_k$ are the coefficients of the polynomial.

For a given matrix $\mathbf{W}$, and a certain degree k, instead of finding weight matrix $\mathbf{W}$ to minimize the spectral radius $\rho(\mathbf{W} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)$, the problem is now to find the polynomial coefficients $\alpha_i, i = 0, \ldots, k$ such that

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in R^{k+1}} & \rho(\sum_{i=0}^{k}\alpha_i\mathbf{W}^i - \frac{1}{N}\mathbf{1}\mathbf{1}^T) \\ \text{subject to} & (\sum_{i=0}^{k}\alpha_i\mathbf{W}^i)\mathbf{1} = \mathbf{1}. \end{array}$$

After obtaining the coefficients α_i, one can run the polynomial filtered distributed consensus proposed in [47]:

Algorithm 2 (*Polynomial filtered distributed consensus*)

(a) Inputs:

- *polynomial coefficients* $\alpha_i, i = 0, \ldots, k+1$, *k degree of the polynomial filter.*
- *tolerance* ε.

(b) *Initialization: Consensus Matrix* $\boldsymbol{W}$, $\boldsymbol{x}(0)$;

(c) *Set t=1;*

(d) ***Repeat***

- ***if*** $mod(t, k+1) == 0$ *then*

$$x_i[t] = \alpha_0 x_i[t-k-1] + \alpha_1 x_i[t-k] + \alpha_2 x_i[t-k+1] + \ldots + \alpha_k x_i[t-1].$$

$$x_i[t] = w_{ii} x_i[t] + \sum_{j \in N_i} w_{ij} x_j[t], \qquad i = 1, 2, \ldots, N$$

else

$$x_i[t] = w_{ii} x_i[t-1] + \sum_{j \in N_i} w_{ij} x_j[t-1]$$

- ***end if***
- $t = t + 1$;
- $\bar{x}[t] = x_i[t]$;

(e) ***until*** $\bar{x}[t] - \bar{x}[t-1] < \varepsilon$.

(f) ***Outputs*** $\bar{x}[t]$

C. Chebyschev polynomials for distributed consensus, [53]:

Denote the Chebyshev polynomial of degree k by $T_k(x)$. These polynomials satisfy:

$$T_k(x) = cos(k \arccos x), \forall x \in [-1, 1],$$

and $|T_k(x)| > 1$ when $|x| > 1, \forall k \in \mathbb{N}$. These polynomials are defined by recurrence:

- $T_0(x) = 1$,
- $T_1(x) = x$,
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), k \geq 2$.

Chebyshev polynomials have the following properties:

- $T_k(x) = cos(k \arccos x) \leq 1$, for $x \in [-1, 1]$.
- $\max_{x \in [-1,1]} |T_k(x)| = 1, T_k(-1) = (-1)^k$ and $|T_k(x)| > 1, \forall |x| > 1$.

On the other hand, the direct expression of $T_k(x)$ is characterized by:

$$T_k(x) = \frac{1 + \tau^{2k}}{2\tau^k}, \tau = \tau(x) = x - \sqrt{x^2 - 1}.$$

In order to speed up the consensus, the main idea consists in designing a distributed linear iteration such that the execution of a fixed number of n steps is equivalent to the evaluation of some polynomial, $P_k(x)$, in the fixed matrix $\mathbf{W}$. The polynomial must satisfy that $P_k(1) = 1$ and $|P_k(x)| < 1$ if $|x| < 1$.

With two real coefficients κ, υ $(-1 < \kappa < \upsilon < 1)$, the polynomial is defined as:

$$P_k(x) = \frac{T_k(cx - d)}{T_k(c - d)}, \text{with } c = \frac{2}{\kappa - \upsilon},\ d = \frac{\kappa + \upsilon}{\kappa - \upsilon}, \forall k$$

which has the following properties:

- if $x \in [\kappa, \upsilon]$, then $cx - d \in [-1, 1]$.
- $P_k(1) = 1$ and $P_k(\kappa + \upsilon - 1) = (-1)^k$.
- $|P_k(x)| < 1, \forall x \in (\kappa + \upsilon - 1, 1)$ and $|P_k(x)| > 1$ otherwise.

The polynomial satisfies the recurrence:

$$P_k(x) = 2\frac{T_{k-1}(c - d)}{T_k(c - d)}(cx - s)P_{k-1}(x) - \frac{T_{k-2}(c - d)}{T_k(c - d)}P_{k-2}(x).$$

The consensus rule is then:

$$\mathbf{x}(k) = P_k(\mathbf{W})\mathbf{x}(0), \tag{1.2.14}$$

where,

$$\mathbf{x}(1) = P_1(\mathbf{W})\mathbf{x}(0) = \frac{1}{T_1(c - d)}(c\mathbf{W} - d\mathbf{I}_N)\mathbf{x}(0),$$

and, $\mathbf{x}(k) = P_k(\mathbf{W})\mathbf{x}(0)$

$$= \left(2\tfrac{T_{k-1}(c-d)}{T_k(c-d)}(c\mathbf{W} - d\mathbf{I}_N)P_{k-1}(\mathbf{W}) - \tfrac{T_{k-2}(c-d)}{T_k(c-d)}(c\mathbf{W} - d\mathbf{I}_N)P_{k-2}(\mathbf{W})\right)\mathbf{x}(0)$$

$$= 2\tfrac{T_{k-1}(c-d)}{T_k(c-d)}(c\mathbf{W} - d\mathbf{I}_N)\mathbf{x}(k-1) - \tfrac{T_{k-2}(c-d)}{T_k(c-d)}(c\mathbf{W} - d\mathbf{I}_N)\mathbf{x}(k-2),\ k \geq 2.$$

The choice of the polynomial determine the convergence speed of the algorithm, given by $\max_{\lambda_i(\mathbf{W})} |P_k(\lambda_i(\mathbf{W}))|$, where $\lambda_i(\mathbf{W})$ is the $i - th$ eigenvalue of $\mathbf{W}$.

The convergence rate is given by:

$$\max_{\lambda_i(\mathbf{W}) \neq 1} \frac{|T_k(c\lambda_i - d)|}{T_k(c - d)} = \max\{\frac{|T_k(c\lambda_N(\mathbf{W}) - d)|}{T_k(c - d)}, \frac{|T_k(c\lambda_2(\mathbf{W}) - d)|}{T_k(c - d)}\}$$

Using the Chebyshev polynomials means that we are minimizing $\max_{\lambda \in [-1,1]} P_k(\lambda)$, therefore, getting high chances to obtain a good convergence rate. By an optimal selection of κ, υ, we can maximize the convergence speed. Following the Theorem 4.2 in [53], the optimal parameters are $\kappa = \lambda_N(\mathbf{W})$ and $\upsilon = \lambda_2(\mathbf{W})$.

Let compare the convergence speed between the standard consensus protocol (1.2.1) and the consensus protocol using the Chebyshev polynomials (1.2.14).

Denoting by $\mathbf{v}_i, \lambda_i(\mathbf{W}), i = 1, \ldots, N$ the eigenvectors and eigenvalues of $\mathbf{W}$, (1.2.1) can be expressed as:

$$\mathbf{x}(k) = \mathbf{W}^k\mathbf{x}(0) = \mathbf{v}_1 + \lambda_2^k(\mathbf{W})\mathbf{v}_2 + \ldots + \lambda_N^k(\mathbf{W})\mathbf{v}_N.$$

Theorem 1.2.3 (*[53]*)

Let $\lambda = \max(|\lambda_2(\boldsymbol{W})|, |\lambda_N(\boldsymbol{W})|)$ be the convergence rate of the standard consensus protocol (1.2.1). For any $0 < \upsilon < \frac{2\lambda}{\lambda^2+1}$ and $\kappa = -\upsilon$, $P_k(\lambda)$ goes to zero faster than λ^k when k goes to infinity. Therefore, the consensus protocol using Chebyshev polynomial converges faster than the standard one.

Since the average consensus protocols can be embedded in more sophisticated distributed algorithms, protocols that guarantee a minimal execution time are more appealing than those ensuring asymptotic convergence. For this purpose, several contributions dedicated to **finite-time consensus** have been recently published in the literature, meaning that consensus is obtained in a finite number of steps. The main contribution of this thesis is to address the design of finite-time average consensus problem by making use of distributed optimization methods.

1.3 Network robustness

The aim of network robustness research is to find a **robustness measure** to evaluate the performance of a network, [16, 21]. Furthermore, understanding whether a network is robust can protect and improve the performance of the network efficiently. By the way, it is also used to design new networks that are able to perform well when facing with failures or attacks.

Several robustness measures have been proposed in the literature. However, we are only interested in topological measures, since it is well known that certain topologies exhibit high robustness against failures. Therefore, they can be used to change the network topology and decrease failures.

1.3.1 Vertex (Edge) connectivity

The **vertex (edge) connectivity** of an incomplete graph $\mathcal{K}_v(\mathcal{K}_e)$ respectively represents the minimal number of vertices (edges) that has to be removed to disconnect the graph. Hence, $\mathcal{K}_v(\mathcal{K}_e)$ depends on the least connected part of the graph. It has been shown in [32] that $\mathcal{K}_v \leq \mathcal{K}_e \leq d_{min}$, where d_{min} is the minimum degree of the network.

For a complete graph of N nodes, $\mathcal{K}_v = \mathcal{K}_e = N - 1$.

Remark 1

- *The measure value $\mathcal{K}_v(\mathcal{K}_e)$ is an integer,*
- *The larger $\mathcal{K}_v(\mathcal{K}_e)$, the harder it is to disconnect the graph, thus the graph is more robust.*
- *In order to compute $\mathcal{K}_v(\mathcal{K}_e)$, the network topology should be known at the first sight.*

1.3.2 Algebraic connectivity

The algebraic connectivity is the second smallest eigenvalue of the Laplacian matrix. If $\lambda_2(\mathbf{L}) = 0$, the graph is disconnected. The algebraic connectivity of an incomplete graph is not greater than the vertex connectivity $\mathcal{K}_v$:

$$0 \leq \lambda_2(\mathbf{L}) \leq \mathcal{K}_v \leq \mathcal{K}_e \leq d_{min}$$

Remark 2

- *A higher value of $\lambda_2(\mathbf{L})$ means that the graph is more robust.*
- *Sometimes, there is a problem that the algebraic connectivity is not strictly increasing when an edge is added, [21].*

1.3.3 Number of spanning trees ξ

A spanning tree is a sub-graph containing $N - 1$ edges, all N vertices, and no cycles. The number of spanning trees is defined as a function of the Laplacian eigenvalues [21]

$$\xi = \frac{1}{N} \prod_{i=2}^{N} \lambda_i(\mathbf{L}) \tag{1.3.1}$$

The number of spanning trees can be used to assess the robustness of the network,[4].

Remark 3

The greater ξ, more robust is the graph.

1.3.4 Effective graph resistance $\mathcal{R}$

Assume the graph is seen as an electrical circuit, where an edge (i, j) corresponds to a resistor of unit resistance $\mathcal{R}_{ij} = 1$ Ohm. Informally, the effective resistance between two vertices of a network (when a voltage is applied across them) can be calculated by series and parallel manipulations. The effective graph resistance is the sum of the effective resistances over all pairs of vertices, [21].

$$\mathcal{R} = \sum_{1 \le i < j \le N} \mathcal{R}_{ij} = N \sum_{i=2}^{N} \frac{1}{\lambda_i(\mathbf{L})} \tag{1.3.2}$$

The effective graph resistance is the sum of the inverse of nonzero Laplacian eigenvalues.

Remark 4

The smaller $\mathcal{R}$, more robust is the graph.

1.4 Outline of this Thesis

For the general picture, the contributions of this thesis can be viewed as three-fold one as shown in the following scheme.

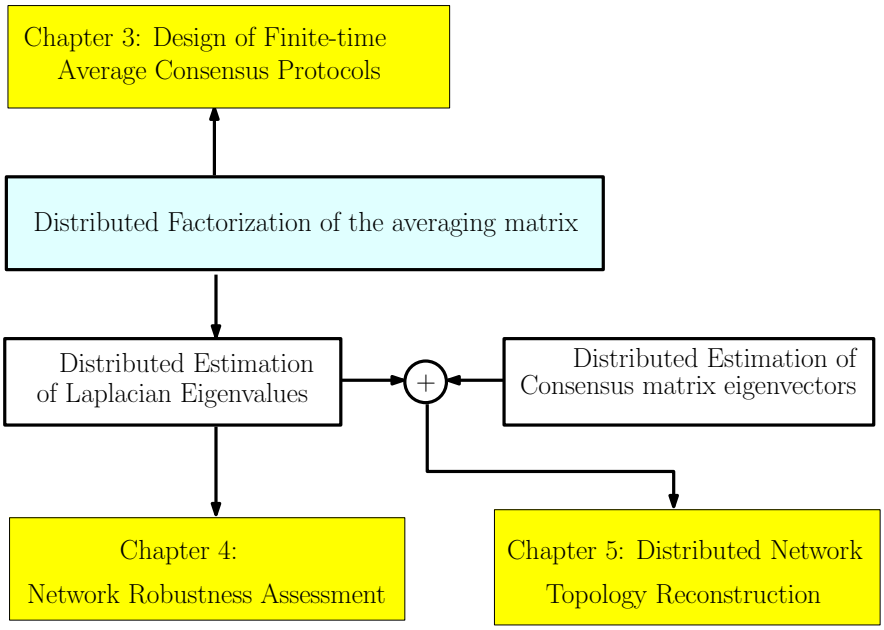


Figure 1.5: Scheme of the thesis

The first task is dedicated to the design of the finite-time average consensus protocols, while the second task focuses on network robustness assessment. The remaining goal is

dedicated to network topology reconstruction. Hereinafter, we present the outline of this thesis as follows:

- Chapter 2: Graph Theory.

  In this chapter, we give general definitions of graph theory.

- Chapter 3: Distributed Design of Finite-time Average Consensus Protocols.

  This is one of the main contributions of this thesis, where the design of the finite-time average consensus is solved in a distributed way. The problem is formulated as a factorization of the averaging matrix $\mathbf{J}_N = \frac{\mathbf{1}\mathbf{1}^T}{N}$. The solution is obtained by solving a constrained optimization problem in a distributed way. For the simple understanding, each linear iteration $k, k = 0, \ldots, D$ with D being the number of steps is illustrated as a layer of a network (including $D + 1$ layers). The main idea is to compute the partial derivatives of the error between the actual output and desired output, then propagate them back to each iteration k in order to update the weights. Then, the proposed algorithm is based on the back-propagation method and an associated gradient descent method. The evaluation of the proposed method is assessed by numerical results.

- Chapter 4: Distributed network robustness assessment.

  In this chapter, we make use of the metrics which are the number of spanning tree ξ and the effective graph resistance $\mathcal{R}$ to assess the robustness of a given network. By the nature of these metrics, they are all functions of the Laplacian eigenvalues $\lambda_i(\mathbf{L})$, hence, the Laplacian eigenvalues estimation is now the main purpose of this task. In this chapter, instead of estimating the Laplacian eigenvalues directly, we solve this problem as a constrained optimization problem with respect to the inverse of the nonzero distinct Laplacian eigenvalues. As the result, by applying the gradient descent method, we finally get the set of nonzero distinct Laplacian eigenvalues. Furthermore, by using an integer programming algorithm, we can estimate the multiplicities corresponding to these eigenvalues. Then, we deduce the whole spectrum of the Laplacian matrix associated with the graph $G(V, E)$. Thanks to these results, the robustness measures can be estimated to evaluate the robustness of the network.

  For the task of Laplacian spectrum estimation, all methods proposed in this chapter are divided into two categories: non-convex and convex formulations. According to non-convex optimization problem (it is also called direct method), the problem is viewed as a constrained optimization problem with respect to the stepsizes α_k of the Laplacian-based consensus protocol, whose inverses are equal to the nonzero distinct Laplacian eigenvalues.

  Solving the non-convex optimization problem, the main issue is the slowness of the convergence speed. Therefore, we try to make the speed of the estimation of

the Laplacian eigenvalues as fast as possible. The main point is to transform the nonconvex problem into a convex one. We solve the problem by two algorithms using the distributed projection method, [57] and the Alternating Direction of Multipliers Method (ADMM), [11]. We consider three cases: final average value perfectly known, noisy and completely unknown.

- Chapter 5: Network topology reconstruction.

  Since, the components used to infer the network structure are the eigenvalues and eigenvectors of the weight consensus matrix, we first propose an algorithm to obtain these eigenvectors. Besides that, the corresponding eigenvalues can be obtained by using the algorithms proposed in the previous chapters. Finally, we can make use of received eigenvectors and eigenvalues to reconstruct the network topology.

- Chapter 6: Conclusion and Future Works.

  This chapter summarises the contributions of this thesis and also outlines possible future directions.

1.5 Publications list

The main contributions of this thesis are the principal subject of the following publications:

1.5.1 International conference papers with proceedings

- Tran, T.M.D; Kibangou, A., "Consensus-based Distributed Estimation of Laplacian Eigenvalues of Undirected Graphs", in 12th European Control Conference (ECC 2013)", Zurich, Switzerland, July 2013.

- Tran, T.M.D; Kibangou, A., "Distributed Design of Finite-time Average Consensus Systems", in IFAC Workshop on Distributed Estimation and Control in Networked (NecSys), Koblenz, Germany, September 2013.

- Tran, T.M.D; Kibangou, A., "Distributed Estimation of Graph Laplacian Eigenvalues by the Alternating Direction of Multipliers Method", in the 19th World Congress of the International Federation of Automatic Control, Cape Town, South Africa, 24-29 August 2014.

1.5.2 Journals

- Tran, T.M.D; Kibangou, A., "Distributed Estimation of Laplacian Eigenvalues of Medium-Size Networks via Constrained Consensus Optimization Problems", in Systems and Control Letters, 2015.

Chapter 2

Graph Theory

Contents

Conceptually, a graph $G(V, E)$ is formed by vertices and edges connecting the vertices. As said in Subsection 1.2, a network can be modelled by a graph, where an agent is represented by a vertex (node) and the communication between agents is set by an edge (link).

According to the communication policy, the graph $G(V, E)$ can be undirected or directed.

- Undirected Graph:

  The communication topology of the network of N agents is represented by a graph $G(V, E)$, where $V = \{1, 2, \ldots, N\}$ is the set of vertices (agents or nodes), and $E \subseteq V \times V$ is the set of edges (links between agents).

  If there is no direction assigned to the edges, then both edges (i, j) and (j, i) are included in the set of edges E. The graph is called *undirected graph*.

  If agent i and j are connected, then the link between i and j is included in E, $(i, j) \in E$ and i and j are called neighbors. The set of neighbors of agent i is denoted by N_i and its degree is denoted by $d_i = |N_i|$, where $|.|$ stands for the cardinality.

- Directed Graph:

If a direction is assigned to the edges, the relations are asymmetric and the graph is called a *directed graph* (or a digraph). For a directed graph as shown in Figure 2.1, for a directed edge (i, j), i is called the head and j is called the tail.

A vertex i is connected to j by a directed edge, or that j is a neighbor of i if $(i, j) \in E$. The edge (i, j) is then an outgoing edge for i and an ingoing edge for j.

The out-degree d_i^{out} of a vertex i is its number of outgoing edges, or number of vertices to which it is connected. The in-degree d_i^{in} is its number of ingoing edges or number of vertices that are connected to it.

$$\sum_{i \in V} d_i^{out} = \sum_{i \in V} d_i^{in} = |E|.$$

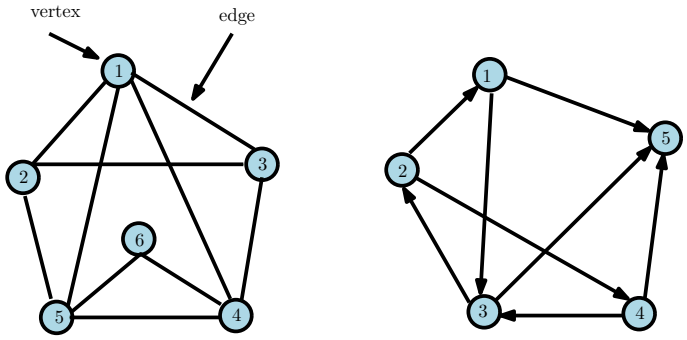


Figure 2.1: 6-node undirected and 5-node directed networks.

Figure 2.1 shows:

- a 6-node undirected graph with set of vertices $V = \{1, 2, \ldots, 6\}$, and set of edges $E = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6), (5, 6)\}$
- a 5-node directed graph with set of vertices $V = \{1, 2, \ldots, 5\}$, and set of edges $E = \{(1, 5), (1, 3), (2, 1), (2, 4), (3, 2), (3, 5), (4, 3), (4, 5)\}$.

2.1 Connectivity of a graph

A *path* from a vertex i to a vertex j is a sequence of distinct vertices starting with vertex i and ending with vertex j such that consecutive vertices are adjacent. A simple path is a path with no repeated vertices.

- In an undirected graph G, two vertices i and j are **connected** if there is a path from i to j. An undirected graph G is connected if for any two vertices in G

there is a path between them, see Figure 2.2. Conversely, two vertices i and j in G are **disconnected** if there is no path from i to j. An undirected graph G is disconnected if we can partition its vertices into two nonempty sets χ and Γ such that no vertex in χ is adjacent to a vertex in Γ, see Figure 2.3.

- A directed graph is **strongly connected** if between every pair of distinct vertices (i, j) in G, there is a directed path that begins at i and ends at j. It is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected undirected graph.

- A graph is said to be **complete (fully-connected)** if every pair of vertices has an edge connecting them, meaning that the number of neighbors of each vertex is equal to $N - 1$, see Figure 2.4.

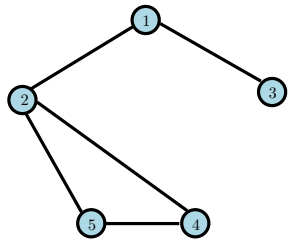


Figure 2.2: A connected undirected graph.

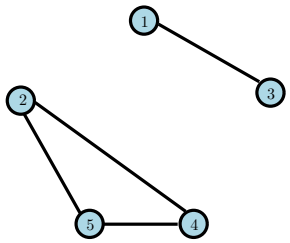


Figure 2.3: A disconnected graph.

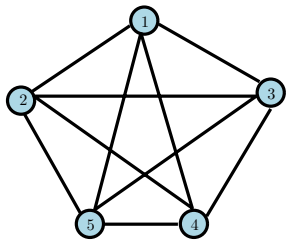


Figure 2.4: A complete graph.

2.2 Algebraic graph properties

- Two vertices joined by an edge are called the endpoints of the edge. If vertex i and vertex j are endpoints of the same edge, then i and j are said to be adjacent to each other. In an undirected graph, vertices that are adjacent to a vertex i are called the neighbors of i. The set of all neighbors of a vertex i is defined as $N_i = \{j \in V : (i, j) \in E\}$.

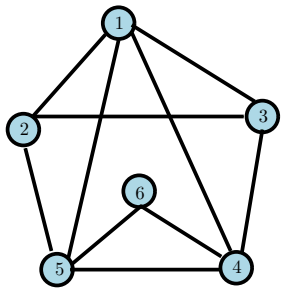
- Given two vertices i and j, the distance $dist(i, j)$ is the length of the shortest simple path between i and j.

- The eccentricity ϵ_i of a vertex i is the greatest distance between i and any other vertex $j \in V$.
- The radius $r(G)$ of a graph G is the minimum eccentricity of any vertex.
- The diameter $d(G)$ of a graph is the maximum eccentricity of any vertex in the graph, $d(G) = \max_{i,j} dist(i,j)$.
- The structure of a graph with N nodes is described by means of an $N \times N$ matrix. The adjacency matrix $\mathbf{A}$ is the matrix with entries $a_{i,j}$ given by

$$a_{ij} = \begin{cases} 1, & \text{if} \quad (i,j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{2.2.1}$$

meaning that, the (i,j) entry of $\mathbf{A}$ is 1 only if vertex j is a neighbor of vertex i. The diagonal entries of $\mathbf{A}$ are all equal to 0. If G is an undirected graph, $a_{ij} = a_{ji}$, i.e., $\mathbf{A}$ is symmetric. In case of a directed graph, $\mathbf{A}$ is asymmetric.

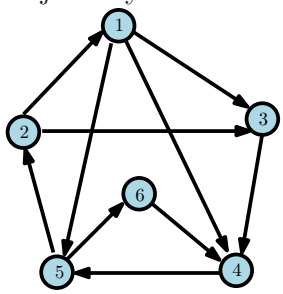
For instance, take the undirected graph in Figure 2.1 as an example, we have the adjacency matrix defined as follows:



Undirected graph

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

An adjacency matrix for a directed graph is defined as follows:



Directed graph

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- The in-degree and out-degree of a vertex i are defined by the sums of the weights of the outgoing and the incoming edges respectively, i.e., $d_i^{in} = \sum_{j=1}^{N} a_{ij}$ and $d_i^{out} = \sum_{j=1}^{N} a_{ji}$. A vertex i is said to be balanced if its in-degree and out-degree are equal, i.e., $d_i^{in} = d_i^{out}$. Therefore, all undirected graphs are balanced graphs.

- A digraph G is called **balanced** if $\sum_{j\neq i}^{N} a_{ij} = \sum_{j\neq i}^{N} a_{ji}$.

  The degree matrix $\mathcal{D}$ of G is the $N \times N$ diagonal matrix with (i, j) entry given by:

$$\mathcal{D}_{ij} = \begin{cases} d_i^{out}, & \text{if} \quad i = j \\ 0, & otherwise \end{cases} \tag{2.2.2}$$

  or equivalently,

$$\mathcal{D} = diag(\mathbf{A1}). \tag{2.2.3}$$

2.3 Spectral graph properties

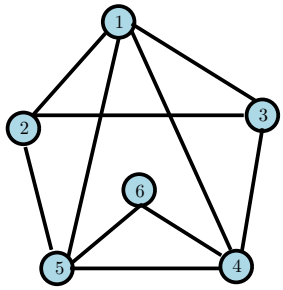
- The Laplacian matrix is used for mathematical convenience to describe the connectivity in a more compact form. The graph Laplacian $\mathbf{L}$ is defined as the matrix with entries l_{ij} given by:

$$l_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{N} a_{ik} & \text{if} \quad i = j \\ -a_{ij} & \text{if} \quad i \neq j \end{cases} \tag{2.3.1}$$

  The Laplacian matrix $\mathbf{L}$ can be expressed in matrix form as follows:

$$\mathbf{L} = \mathcal{D} - \mathbf{A}. \tag{2.3.2}$$

  Again, we take the 6-node graph in Figure 2.1 as an example:



Undirected graph

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 \\ -1 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

- Some important properties of the Laplacian matrix are:

  1. As can be seen in the definition of $\mathbf{L}$, the row-sum of $\mathbf{L}$ is zero. Since $\mathbf{L1} = \mathbf{0}$, where $\mathbf{0}$ is a zero vector of length N, $\mathbf{L}$ has at least one eigenvalue zero with associated right eigenvector $\mathbf{1}$.

     If the graph $G(V, E)$ is undirected, then:

2. The spectrum of $\mathbf{L}$, that is the set of all eigenvalues of the Laplacian matrix, is denoted as $sp(\mathbf{L}) = \{\lambda_1^{m_1}, \lambda_2^{m_2}, \ldots, \lambda_{D+1}^{m_{D+1}}, \}$, where m_i stands for the multiplicity of the $i-th$ eigenvalue and D is the number of nonzero distinct Laplacian eigenvalues. The multiplicity of the zero eigenvalue represents the number of connected components of the graph. For undirected graphs, $\mathbf{L}$ is symmetric and positive semi-definite, hence $0 = \lambda_1 < \lambda_2 < \ldots < \lambda_{D+1} < 2d_{max}$, and $\sum_{i=1}^{D+1} m_i = N$ with $m_1 = 1$, where d_{max} is the maximum degree of the graph, i.e., $d_{max} = \max_i d_i$. We denote by $\Lambda = \{\lambda_2, \ldots, \lambda_{D+1}\}$ the set of nonzero distinct Laplacian eigenvalues.

3. λ_2 is the second smallest eigenvalue, which is known as the algebraic connectivity and reflects the degree of connectivity of the graph G, [28]. It is well-known that the speed of convergence of a consensus protocol depends on this second smallest Laplacian eigenvalue λ_2, [32].

$$\min_{x \neq 0, \mathbf{1}^T\mathbf{x}=\mathbf{0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\|\mathbf{x}\|^2} = \lambda_2(\mathbf{L}). \tag{2.3.3}$$

From the literature, the second smallest Laplacian eigenvalue is upper bounded by $\max_{j \in N_i}\{d_i + d_j\}$.

4. In [51], another interesting property has been pointed out:

$$\sum_{i=1}^{N} \lambda_i(\mathbf{L}) = \sum_{i=1}^{N} d_i.$$

2.4 Standard classes of graphs

1. A graph is called weighted if a weight is associated with every edge according to a proper map $\mathbf{W} : E \to R$, such that if $(i, j) \in E$, then $w_{ij} \neq 0$, otherwise $w_{ij} = 0$. In other words, a weighted graph is a graph whose edges have been labelled with real numbers. The length of a path in a *weighted graph* is the sum of the weights of the edges in the path.

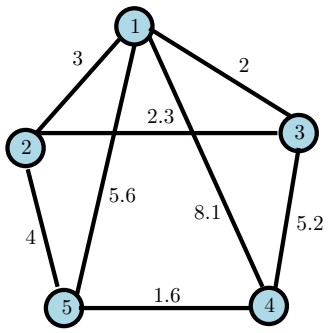


Figure 2.5: A weighted graph.

2. A graph is called a *regular graph* if all the vertices have the same number of neighbors. The graph is written as n-regular if the number of neighboring vertices is n for all vertices. n is called valency of the n-regular graph.

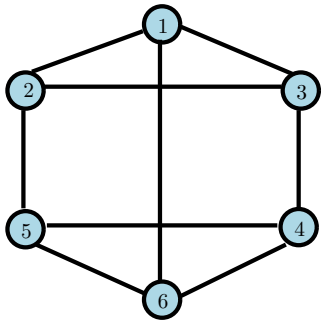


Figure 2.6: A 3-regular graph

3. A graph $G(V, E)$ with diameter $d(G)$ is said to be a *distance-regular graph* if for any vertices i and j of V and any integers $v, w = 0, 1, \ldots, d(G)$ (where $d(G)$ is diameter of graph G), the number of vertices at distance v from i and distance w from j depends only on v, w, and the graph distance between i and j, independently of the choice of i and j.

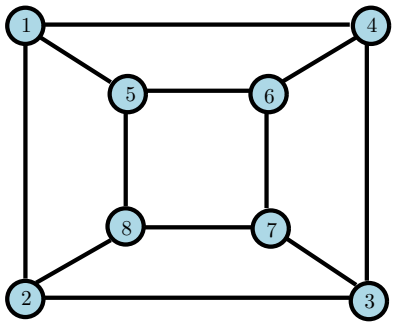


Figure 2.7: A distance-regular graph

4. A graph G is said to be *strongly regular*, SRG(N,κ,a,c), if it is neither complete nor empty and there are integers κ, a, and c such that:

   - G is regular with valency κ;
   - any two adjacent vertices have exactly a common neighbors;
   - any two distinct non-adjacent vertices have exactly c common neighbors.

   These parameters are linked as follows:

$$(N - \kappa - 1)c = \kappa(\kappa - a - 1). \tag{2.4.1}$$

   Examples of strongly regular graphs are:

- Petersen graph ($SRG(10, 3, 0, 1)$),
- Clebsch graph ($SRG(16, 5, 0, 2)$),
- Payley graphs ($SRG(q, \frac{q-1}{2}, \frac{q-5}{4}, \frac{q-1}{4})$, with q congruent 1 (mod 4)),
- n-dimensional Hamming graphs ($SRG(n^2, 2n-2, n-2, 2)$, $n \geq 2$),
- Cycle graphs with $N < 6$.

The most obvious property of the strongly regular graphs is that the diameter is equal to two ($d(G) = 2$).

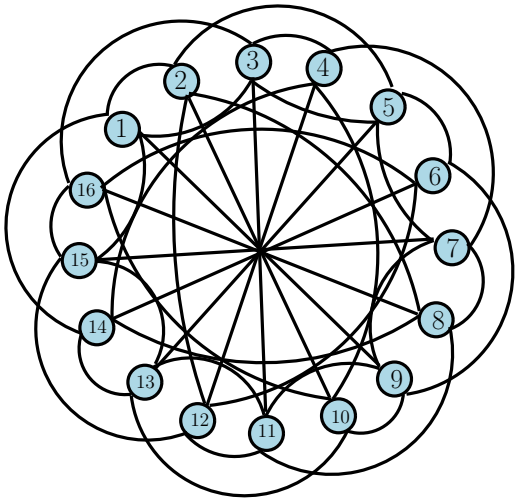


Figure 2.8: A 5-regular Clebsch graph ($SRG(16, 5, 0, 2)$)

Chapter 3

Distributed design of finite-time average consensus protocols

Contents

3.1 Introduction

Consider an undirected connected graph $G(V, E)$ with $x_i(k)$ being the state of node i at time-step k and define the initial state of the given network as $\mathbf{x}(0) = [x_1(0), x_2(0), \ldots, x_N(0)]^T$ where $N = |V|$. A novel method is proposed in this chapter to design finite-time average consensus protocols for MASs or WSNs in a distributed way. In the discrete-time framework, a linear iteration scheme expresses its benefits for a distributed algorithm since each node repeatedly updates its value as a weighted linear

combination of its own value and those of its neighbors.

$$x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), \tag{3.1.1}$$

where

$$w_{ij} = \begin{cases} 0, & \text{if } (i,j) \notin E \\ w_{ij}, & \text{if } (i,j) \in E \end{cases}$$

or equivalently in matrix form:

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k), \tag{3.1.2}$$

$\mathbf{W}(k) \in \mathcal{S}_G$ where $\mathcal{S}_G$ is the set of matrices that can be factorized as $\mathbf{W}(k) = \mathbf{Q}(k) \circ (\mathbf{I}_N + \mathbf{A})$ where $\mathbf{Q}(k)$ stands for an arbitrary square matrix, $\mathbf{I}_N$ as the $N \times N$ identity matrix, and $\circ$ denotes the Hadamard matrix product that corresponds to an entry-wise matrix product.

As presented in the Chapter 1.2, average consensus is reached asymptotically if

$$\lim_{k \to \infty} \mathbf{x}(k) = \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}(0), \tag{3.1.3}$$

meaning that

$$\lim_{k \to \infty} \mathbf{W}^k = \frac{1}{N}\mathbf{1}\mathbf{1}^T = \mathbf{J}_N. \tag{3.1.4}$$

However, the foremost thing to note is that in order to run an average consensus algorithm, two main steps are required: the configuration step (also called design step) and the execution step. During the configuration step, a task can be achieved through a self-configuration algorithm instead of resorting to a network manager. Self-configuration can include graph discovering and distributed decision on some parameters. For instance, if the protocol is the maximum degree weights one (1.2.6), each agent first computes the number of its neighbors before running a max-consensus algorithm for computing the maximum degree of the underlying graph. In the case of the Metropolis-Hasting based protocol (1.2.7), each agent compares its degree with that of its neighbors in order to compute the weights of the average consensus protocol. One commonly used protocol is the constant edge weights, or graph Laplacian based average consensus protocol (1.2.8), where a common stepsize is used by all agents. Asymptotic convergence is hence guaranteed if the stepsize is strictly positive and lower than $\frac{2}{\lambda_N}$, where λ_N stands for the largest Laplacian eigenvalue. Even through there are some simple bounds that give choices for the stepsize without requiring exact knowledge of the Laplacian spectrum, agents have to agree on an adequate stepsize.

To the best of our knowledge, there is no published work dealing with self-configuration protocols for the constant edge weights based average consensus protocol.

In real systems, the execution time is getting more and more impact. Therefore, the purpose is now to design a finite-time average consensus algorithm allowing all nodes to reach the average consensus value in a finite number of steps D for self-configuration protocols, i.e.

$$\mathbf{x}(D) = \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}(0) = \mathbf{J}_N\mathbf{x}(0). \tag{3.1.5}$$

3.2 Literature review

In the literature, the proposed finite-time consensus protocols can be classified by two main approaches, namely: the minimal polynomial concept based approach and matrix factorization based approach.

3.2.1 Minimal polynomial concept based approach

In [70], a finite-time consensus algorithm for linear time-invariant topologies based on the minimal polynomial of the weight consensus matrix $\mathbf{W}$ was proposed. Specifically, the minimal polynomial of the matrix $\mathbf{W}$ is the unique monic polynomial $q(\mathbf{W})$ of the smallest degree such that $q(\mathbf{W}) = 0$.

Suppose that the minimal polynomial of $\mathbf{W}$ has degree $D+1 \leq N$. For some constants $\alpha_0, \alpha_1, \ldots, \alpha_D$, which are coefficients of the minimal polynomial $q(\mathbf{W})$, since $q(\mathbf{W}) = 0$, we get:

$$\mathbf{W}^{D+1} + \alpha_D\mathbf{W}^D + \ldots + \alpha_1\mathbf{W} + \alpha_0\mathbf{I}_N = \mathbf{0}.$$

By multiplying the two sides of this equation with $\mathbf{x}(k)$, we obtain:

$$\mathbf{W}^{D+1}\mathbf{x}(k) + \alpha_D\mathbf{W}^D\mathbf{x}(k) + \ldots + \alpha_1\mathbf{W}\mathbf{x}(k) + \alpha_0\mathbf{x}(k) = \mathbf{0}.$$

or equivalently:

$$\mathbf{x}(k+D+1) + \alpha_D\mathbf{x}(k+D) + \ldots + \alpha_1\mathbf{x}(k+1) + \alpha_0\mathbf{x}(k) = \mathbf{0}.$$

Therefore, $\forall k \geq 0$ and $1 \leq i \leq N$, $x_i(k)$ satisfies a linear difference equation of the form:

$$x_i(k+D+1) + \alpha_D x_i(k+D) + \ldots + \alpha_1 x_i(k+1) + \alpha_0 x_i(k) = 0. \tag{3.2.1}$$

It was shown that:

$$\bar{x}_i = \frac{[x_i(D)\ x_i(D-1)\ \ldots\ x_i(0)]\mathbf{s}}{[1\ 1\ \ldots\ 1]\mathbf{s}},$$

where

$$\mathbf{s} = \begin{bmatrix} 1 \\ 1+\alpha_D \\ 1+\alpha_{D-1} \\ \vdots \\ 1+\sum_{j=1}^{D} \alpha_j \end{bmatrix}$$

The nodes need a sufficient observation time which is the degree of the minimal polynomial to reconstruct the initial states. However, the design of this protocol requires a strong knowledge on the underlying network topology. In fact, each node needs to store $N(N-1)$ samples with N being the network size. Therefore, a decentralized calculation of the minimal polynomial was proposed although it is not entirely local, in the sense that a local calculation is repeated over N independent iterations (where N is the total number of nodes of the network) and at each iteration, it requires each node to store its own values for $N+1$ time steps in order to obtain the minimal polynomial.

An improved method which requires much less data storage has been proposed in [78]. The authors have proposed algorithms to compute the consensus value using the minimal number of observations of an arbitrary chosen node in a network using $2(D_i+1)$ successive discrete-time steps, where D_i+1 is the degree of the monic minimal polynomial.

From equation (3.2.1), the measurable value at node i is determined by the minimal of polynomial of the corresponding matrix observability pair $[\mathbf{W}, \mathbf{e}_i^T]$, where $\mathbf{e}_i^T = [0,\ldots,0,1_i,0,\ldots,0] \in I\!R^{1\times N}$.

Definition 1 (*[78]*)

The minimal polynomial associated with the matrix pair $[\boldsymbol{W}, \boldsymbol{e}_i^T]$ *denoted by* $q_i(t) \triangleq t^{D_i+1} + \sum_{k=0}^{D_i} \alpha_{i,k} t^k$ *is the unique monic polynomial of minimal degree* D_i+1 *that satisfies* $\boldsymbol{e}_i^T q_i(\boldsymbol{W}) = 0$.

Hence, the minimal integer value D_i+1 necessary for (3.2.1) to hold for almost any initial condition $\mathbf{x}(0)$ is given by the degree of the minimal polynomial of the observability pair $[\mathbf{W}, \mathbf{e}_i^T]$ [79].

Consider the vector of difference between successive discrete-time values at node i, $\bar{\mathbf{X}}_i(0,1,2\ldots,2k) = \{x_i(1)-x_i(0), x_i(2)-x_i(1),\ldots,x_i(2k+1)-x_i(2k)\}, k \in \mathbb{Z}$ and its associated Hankel matrix:

$$\Gamma\{\bar{\mathbf{X}}_i(0,1,2\ldots,2k)\} \triangleq \begin{bmatrix} x_i(1)-x_i(0) & x_i(2)-x_i(1) & \cdots & x_i(k+1)-x_i(k) \\ x_i(2)-x_i(1) & x_i(3)-x_i(2) & \cdots & x_i(k+2)-x_i(k+1) \\ \vdots & \vdots & \ddots & \vdots \\ x_i(k+1)-x_i(k) & x_i(k+2)-x_i(k+1) & \cdots & x_i(2k+1)-x_i(2k) \end{bmatrix}. \tag{3.2.2}$$

The minimal number of steps is computed by checking a rank condition of a Hankel matrix constructed from local output observations in a distributed way.

Increase the dimension k of the square Hankel matrix $\Gamma\{\bar{\mathbf{X}}_i(0,1,2\ldots,2k)\}$ until it loses rank and store the first defective Hankel matrix. In [78], the authors have shown that by using a Vandermonde factorization of the Hankel matrix, one can see that $\Gamma\{\bar{\mathbf{X}}_i(0,1,2\ldots,2k)\}$ is defective if $k \geq D_i + 1$.

Then, the normalized kernel $\boldsymbol{\beta} = [\beta_0 \ldots \beta_{D_i+1}\ 1]^T$ of the first defective Hankel matrix is estimated.

Based on linear iteration (3.2.1), the following regression equation denoted by the output observations of node i in minimal time is defined as:

$$x_i(k+D_i+1) + \alpha_{i,D_i}x_i(k+D_i) + \ldots + \alpha_{i,1}x_i(k+1) + \alpha_{i,0}x_i(k) = 0. \tag{3.2.3}$$

Consider the Z-transform of $x_i(k)$:

$$X(z) = \frac{\sum_{j=1}^{D_i+1}\alpha_j \sum_{l=0}^{j-1} x_i(l)z^{j-l}}{q_i(z)} \triangleq \frac{H(z)}{q_i(z)}.$$

In order to estimate the final consensus value at node i, we define the following polynomial:

$$p_i(z) \triangleq \frac{q_i(z)}{z-1} \triangleq \sum_{j=0}^{D_i}\beta_j z^j.$$

Then, the final consensus value is defined:

$$\phi = \lim_{z\to 1}(z-1)X(z) = \frac{H(1)}{p_i(1)} = \frac{\mathbf{y}_i^T(D_i)\boldsymbol{\beta}}{\mathbf{1}^T\boldsymbol{\beta}},$$

where $\mathbf{y}_i^T(D_i) = [x_i(0)\ x_i(1)\ \ldots\ x_i(D_i)]$ and $\boldsymbol{\beta} = [\beta_0\ \ldots\ \beta_{D_i-1}\ 1]^T$ is the vector of coefficients of the polynomial $p_i(z)$.

In [78], the authors show that for an arbitrary initial condition, except for a set of initial conditions with Lebesgue measure zero [7], the consensus value can be obtained from local observations in a minimal number of steps that does not depend on the total size of the graph. However, the computation of the rank of a given matrix and that of its kernel are needed. Therefore, the computational cost is still the weakness of these methods.

3.2.2 Matrix factorization based approach

This approach is based on the factorization of the averaging matrix $\mathbf{J}_N$. More precisely, we desire to find a finite sequence of matrices $\{\mathbf{W}(1), \mathbf{W}(2), \ldots, \mathbf{W}(D)\}, \mathbf{W}(k) \in \mathcal{S}_G$

such that

$$\mathbf{x}(D) = \prod_{k=D}^{1} \mathbf{W}(k)\mathbf{x}(0) = \mathbf{J}_N \mathbf{x}(0) \qquad \text{for all } \mathbf{x}(0) \in I\!R^N, \tag{3.2.4}$$

with

$$\prod_{k=D}^{1} \mathbf{W}(k) = \mathbf{J}_N. \tag{3.2.5}$$

Finding this set of matrices is equivalent to solve a multi-variable polynomial system of equations, which can be solved using Grobner basis theory. However, the computational complexity is also burdens. In general, studying the existence of solutions to such a system of equations can be intractable.

3.2.2.1 Existence

The existence issue has been deeply considered in [30] and [34]. It has been pointed out that no solution exists if the factor matrices $\mathbf{W}(k)$ are all equal except if the graph is complete. Since the diameter of the graph characterizes the time necessary for a given information to reach all the nodes in the network, the number of factor matrices cannot be lower than the diameter $d(G)$.

In [34], an upper-bound of the number of factor matrices is given by $2r(G)$, with $r(G)$ being the radius of the given graph (trees and graphs with minimum diameter spanning tree). Furthermore, for more general graphs, according to the results in [42], the number D is upper bounded by $N-1$. Hence, the number of steps D is bounded by:

$$d(G) \leq D \leq N - 1. \tag{3.2.6}$$

The solution of the factorization problem is not unique and no distributed solution exists.

In [35], the authors have investigated the possibility of reaching (average) consensus in finite time by making use of a linear iterations scheme with a finite sequence of T consensus matrices, which are restricted to be stochastic with positive diagonals and consistent with a connected undirected graph structure $G(V, E)$ where $N = |V|$. In this paper, the upper bound on the number of matrices was also given by $\frac{N(N-1)}{2}$. On the other hand, for directed graphs, the authors have proven that finite-time consensus is reached by a sequence of stochastic matrices with positive diagonals only if the graph is strongly connected and contains a simple directed cycle with even length. However, if the graph G is a simple directed cycle, then there is no sequence of stochastic matrices with positive diagonals achieving consensus.

3.2.2.2 Solutions for particular graphs

For distance regular graphs, a closed form expression of the matrix factorization exists [39]. The factorization of the averaging matrix has been proposed based on parameters of the intersection array characterizing the underlying graph constrained as a distance regular graph.

Definition 2 ([12])

A graph $G(V, E)$ with diameter $d(G)$ is said to be distance regular if there exist integers $a_k, b_k, c_k, k = 0, 1, \ldots, d(G)$ such that for any two vertices $i, j \in V$ and distance $k = dist(i, j)$, there are exactly a_k neighbors of j in $V_k(i)$, b_k neighbors of j in $V_{k+1}(i)$, and c_k neighbors of j in $V_{k-1}(i)$, where $V_k(i)$ is the set of vertices j of $G(V, E)$ with $dist(i, j) = k$.

The parameters of a distance regular graph with valency K are linked as $a_i + b_i + c_i = K$, with $a_0 = c_0 = b_{d(G)} = 0$ and $c_1 = 1$. Therefore, it is usual to characterize a distance regular graph by its intersection array $\{b_0, b_1, \ldots, b_{d(G)}; c_1, c_2, \ldots, c_{d(G)}\}$ where $b_0 \geq b_1 \geq \ldots \geq b_{d(G)}$ and $0 \leq c_1 \leq c_2 \leq \ldots \leq c_{d(g)}$.

Let us define $G_k(V, E_k)$ with $(i, j) \in E_k$ if and only if $dist(i, j) = k$ and $\mathbf{A}_k$ be its adjacency matrix.

$$\sum_{k=0}^{d(G)} \mathbf{A}_k = \mathbf{J}_N.$$

One property can be noted as follows:

$$\mathbf{A}\mathbf{A}_k = b_{k-1}\mathbf{A}_{k-1} + a_k\mathbf{A}_k + c_{k+1}\mathbf{A}_{k+1},$$

where $\mathbf{A} = \mathbf{A}_1$ stands for the graph adjacency matrix.

Example : The Hamming graph $H(d(H), n)$ with $d(H)$ being the diameter of the graph, and $n^{d(H)}$ being the number of vertices.

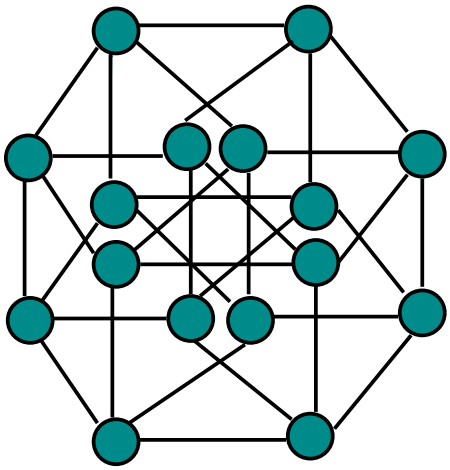


Figure 3.1: A $H(4,2)$ Hamming graph.

Therefore, its intersection array is given by $\{4,3,2,1;1,2,3,4\}$.

Additionally, in [39], the author has presented the factorization of the averaging matrix for distance regular graphs. Herein, we review two main theorems that help to determine the sequence of the consensus matrices.

Theorem 3.2.1 (*[36]*)

Consider a graph G with N vertices and adjacency matrix $\boldsymbol{A}$

1. *There exist a polynomial $\mathcal{H}(\nu)$ such that $\mathcal{H}(\boldsymbol{A}) = \boldsymbol{J}_N$ if and only if G is a connected regular graphs.*
2. *For a connected regular graph with valency K, the unique polynomial of least degree satisfying $\mathcal{H}(\boldsymbol{A}) = \boldsymbol{J}_N$ is $\mathcal{H}(\nu) = \frac{N}{g(K)} g(\nu)$ where $(\nu - K)g(\nu)$ is the minimal polynomial of $\boldsymbol{A}$.*

The polynomial $\mathcal{H}(\mathbf{A})$ is called the Hoffman polynomial of G. We can note that the degree of this polynomial is unknown, except for distance regular graph.

Lemma 1 (*[39]*)

Consider a distance regular graph $G(V,E)$ with $|V| = N$, valency K, diameter $d(G)$ and intersection array $\{b_0, b_1, \ldots, b_{d(G)}; c_1, c_2, \ldots, c_{d(G)}\}$. The averaging matrix $\frac{1}{N}\boldsymbol{J}_N$ is a $d(G)$-th order polynomial of the graph adjacency matrix $\boldsymbol{A}$:

$$\frac{1}{N}\boldsymbol{J}_N = \mathcal{P}_G(\boldsymbol{A}) = \sum_{q=0}^{d(G)} \gamma_q \boldsymbol{A}^q,$$

where $\gamma_q = \sum_{k=0}^{d(G)} \frac{\beta_{k,q}}{N}$, *and the coefficients* $\beta_{k,q}$ *depend on the parameters of the intersection array as follows:*

$$\begin{aligned}
\beta_{k+1,0} &= -\frac{b_{k-1}\beta_{k-1,0} + a_k\beta_{k,0}}{c_{k+1}}, \\
\beta_{k+1,q} &= \frac{\beta_{k,q-1} - b_{k-1}\beta_{k-1,q} + a_k\beta_{k,q}}{c_{k+1}}, \quad 1 \leq q-1, \\
\beta_{k+1,k} &= \frac{\beta_{k,k-1} - a_k\beta_{k,k}}{c_{k+1}}, \\
\beta_{k+1,k+1} &= \frac{\beta_{k,k}}{c_{k+1}},
\end{aligned} \tag{3.2.7}$$

with $a_k = K - b_k - c_k$, $\beta_{0,0} = 1$ *and* $\beta_{k,q} = 0$ *if* $k < q$.

Here, $\mathcal{P}_G(\mathbf{A})$ stands for the polynomial of $\mathbf{A}$. Since $\mathcal{P}_G(\mathbf{A}) = \frac{1}{N}\mathcal{H}(\mathbf{A})$, the polynomial $\mathcal{P}_G(\nu) = \sum_{q=0}^{d(G)} \gamma_q \nu^q$ will be called the averaging Hoffman polynomial of G ($\mathcal{P}_G(K) = \frac{1}{N}\mathcal{H}(K) = 1$). The roots of $\mathcal{P}_G(\nu)$ are also those of the minimal polynomial of $\mathbf{A}$.

Theorem 3.2.2 (*[39]*)

Let $\mathcal{P}_G$ *be the averaging Hoffman polynomial of the distance regular graph* $G(V, E)$ *of valency* K, *diameter* $d(G)$, *adjacency matrix* $\boldsymbol{A}$, *and* $|V| = N$. *The averaging matrix* $\frac{1}{N}\boldsymbol{J}_N$ *can be factorized as:*

$$\frac{1}{N}\boldsymbol{J}_N = \prod_{t=1}^{d(G)} ((1 - \alpha_t K)\boldsymbol{I}_N + \alpha_t \boldsymbol{A}), \qquad \alpha_t = \frac{1}{K - \epsilon_t}, \tag{3.2.8}$$

where $\epsilon_t \in \mathbb{R} \setminus \{K\}$ *stands for the roots of* $\mathcal{P}_G$.

Let take the Hamming graph $H(4, 2)$ depicted in Figure 3.1 with valency $K = 4$, diameter $d(H) = 4$ and $N = 2^4 = 16$. From (3.2.7), we obtain the coefficients as follows:

$$\begin{aligned}
\beta_{0,0} &= 1; \quad \beta_{1,0} = 0; \quad \beta_{1,1} = 1; \quad \beta_{2,0} = -2; \quad \beta_{2,1} = 0; \quad \beta_{2,2} = \frac{1}{2}; \quad \beta_{3,0} = 0 \\
\beta_{3,1} &= -\frac{5}{3}; \quad \beta_{3,2} = 0; \quad \beta_{3,3} = \frac{1}{6}; \quad \beta_{4,0} = 1; \quad \beta_{4,1} = 0; \quad \beta_{4,2} = -\frac{2}{3}; \quad \beta_{4,3} = 0; \\
\beta_{4,4} &= \frac{1}{24}.
\end{aligned}$$

As the result, we compute $\gamma_q = \sum_{k=0}^{d(H)} \frac{\beta_{k,q}}{N}$ as follows:

$$\gamma_0 = 0; \quad \gamma_1 = -\frac{2}{3 \times N}; \quad \gamma_2 = -\frac{1}{6 \times N}; \quad \gamma_3 = \frac{1}{6 \times N}; \quad \gamma_4 = \frac{1}{24 \times N}.$$

Hence, the averaging polynomial is given by $\mathcal{P}_H(\nu) = \frac{1}{N}(-\frac{2}{3}\nu - \frac{1}{6}\nu^2 + \frac{1}{6}\nu^3 + \frac{1}{24}\nu^4)$. Solving this polynomial, we get the roots $\epsilon = \{-4, -2, 0, 2\}$. Furthermore, from (3.2.8), the corresponding stepsizes are obtained as $\alpha = \{\frac{1}{8}, \frac{1}{6}, \frac{1}{4}, \frac{1}{2}\}$, which are exactly the nonzero inverses of the Laplacian eigenvalues of the graph. Therefore, the sequence of the consensus matrices are:

$$\begin{aligned} \mathbf{W}(1) &= \frac{1}{2}\mathbf{I}_N - \frac{1}{8}\mathbf{A}; \quad \mathbf{W}(2) = \frac{1}{3}\mathbf{I}_N - \frac{1}{6}\mathbf{A}; \\ \mathbf{W}(3) &= \mathbf{A}; \qquad\qquad \mathbf{W}(4) = -\mathbf{I}_N + \frac{1}{2}\mathbf{A}. \end{aligned}$$

3.2.2.3 General solutions

In [46], the resulting solution yields a link scheduling on the complete graph to achieve finite-time consensus. Such scheduling is to be controlled by a central node. They provide necessary and sufficient conditions for finite-time consensus and compute the minimum consensus time on the Boolean hypercube. In order to prove the existence of a finite factorization of $\mathbf{J}_N$, the mechanism is that starting with a spanning tree T of graph G and arbitrarily designate a node as the root. The basic idea is that all nodes pass all their goods to the chosen root. Then, the root propagates the appropriate amount of goods back into the network so that every node has an equal amount at the end. The proposed method includes two actions. The first action is described that from vertices farthest from the root of a spanning tree T, the proposed algorithm traverses upwards to give all goods to their parents (they can be children of other parents), then remove themself. This action terminates when only a single vertex (the root) remains. At this point, the remain node (the root) contains the sum of all initial values. Then, the second action is to propagate back down the tree while re-distributing the values to achieve the average consensus value at termination.

For more general graphs, solutions based on graph Laplacian have been recently introduced in closed-form, [41, 42].

Lemma 2 (*[41]*)

The matrices $\boldsymbol{W}(k) = \alpha_k \boldsymbol{I}_N + \beta \hat{\boldsymbol{A}}, k = 1, \ldots, D$ are jointly diagonalizable, consistent with the network topology, and admit $\boldsymbol{1}$ as eigenvector.

Where, $\hat{\mathbf{A}}$ is defined as $\hat{\mathbf{A}} = d_{max}\mathbf{I}_N - \mathbf{L}$ whose properties are:

(a) $\hat{\mathbf{A}}$ is symmetric;

(b) Its eigenvalues $\lambda_k(\hat{\mathbf{A}}) = d_{max} - \lambda_k(\mathbf{L})$, where $\lambda_k(\mathbf{L}), d_{max}$ are the eigenvalues of the Laplacian matrix $\mathbf{L}$ and the maximum degree respectively. In particular, d_{max} is a simple eigenvalue.

(c) The eigenvectors of the Laplacian matrix are also the eigenvectors of $\hat{\mathbf{A}}$. Meaning that, $\mathbf{1}$ is the eigenvector associated with the eigenvalue d_{max}.

Theorem 3.2.3 ([41])

Given a connected undirected graph associated with the Laplacian matrix $\boldsymbol{L}$, the set of matrices $\boldsymbol{W}(k) = (\alpha_k + d_{max}\beta)\boldsymbol{I}_N - \beta\boldsymbol{L}, k = 1, \ldots, D, \beta \neq 0$, allows reaching the average consensus in D steps if:

(a) $D+1$ is the number of distinct eigenvalues of the Laplacian matrix;

(b) the parameter β and α_k are defined as:

$$\begin{aligned} \beta &= \frac{1}{\prod_{i=2}^{D+1} \sqrt[D]{\lambda_i(\boldsymbol{L})}} \\ \alpha_k &= \frac{\lambda_{k+1}(\boldsymbol{L}) - d_{max}}{\prod_{i=2}^{D+1} \sqrt[D]{\lambda_i(\boldsymbol{L})}}, \quad k = 1, \ldots, D. \end{aligned}$$

with $\lambda_i(\boldsymbol{L}), i = 2, \ldots, D+1$ being the nonzero distinct Laplacian eigenvalues.

More precisely, in [42], the solution was given by $\alpha_k = 1$ and $\beta_k = -\frac{1}{\lambda_{k+1}(\mathbf{L})}$, $\lambda_k(\mathbf{L})$ being a nonzero Laplacian eigenvalue.

Remark 5

- *The number of factor matrices is the number of distinct nonzero Laplacian eigenvalues.*
- *The proposed solutions make use of the Laplacian spectrum. For this purpose, we propose distributed methods for Laplacian spectrum estimation in Chapter 4.*

3.2.3 Comparison between the minimal polynomial approach and the matrix factorization approach

To summarise the difference between the two approaches, we focus on two terms that are design time and execution time.

- Design time:
  (a) For the minimum polynomial approach, each node chooses weights to generate the weight consensus matrix $\mathbf{W}$ that satisfies the convergence conditions 1.2.2.

(b) The matrix factorization approach requires the storage and computation resources for global information (network topology or Laplacian spectrum) to design the weight consensus matrix $\mathbf{W}$.

- Execution time:

(a) The minimum polynomial approach mainly needs data storage and computation resources: $2(D_r + 1)$ successive steps, [78].

(b) At this stage, the factorization of averaging matrix does not require storage of past data. Hence, the execution time is as simple as a standard consensus protocol.

The purpose of this thesis is to study a new algorithm for self-configuration for finite-time average consensus where the weight matrices are not necessarily based on the graph Laplacian. The aim is to design protocols that allow achieving average consensus in the fastest possible time, possibly as fast as diameter of the underlying graph. More precisely, we solve a matrix factorization problem in a distributed way.

3.3 Distributed solution to the matrix factorization problem.

For a connected graph $G(V, E)$, let $\mathbf{x}(0) = [x_1(0), x_2(0), \ldots, x_N(0)]^T$ be the state vector of the $N-$agent network associated with the graph G. For the sake of simplicity, N agents (nodes) of the network (graph G) are arranged in the first layer that corresponds to the initial state $\mathbf{x}(0)$ as described in the Figure 3.2 illustrating the linear iteration scheme (3.1.2) in space and time.

As presented in Chapter 1.2, the linear iteration scheme is an useful tool that allows the system to reach the finite-time average consensus in D steps. Therefore, according to each iteration, the updated state vectors $\mathbf{x}(k), k = 1, \ldots, D-1$ are assigned to be consecutive layers respectively. Finally, the last iteration is dedicated to the output values $\mathbf{x}(D)$. The weight matrices $\mathbf{W}(k), k = 1, \ldots, D$ have the entries $w_{ij}(k)$ are correspondent to the links between the nodes of two consecutive layers at time-step k. For instance, $\mathbf{W}(1)$ is the weight matrix assigned to the link between the first layer and the second one.

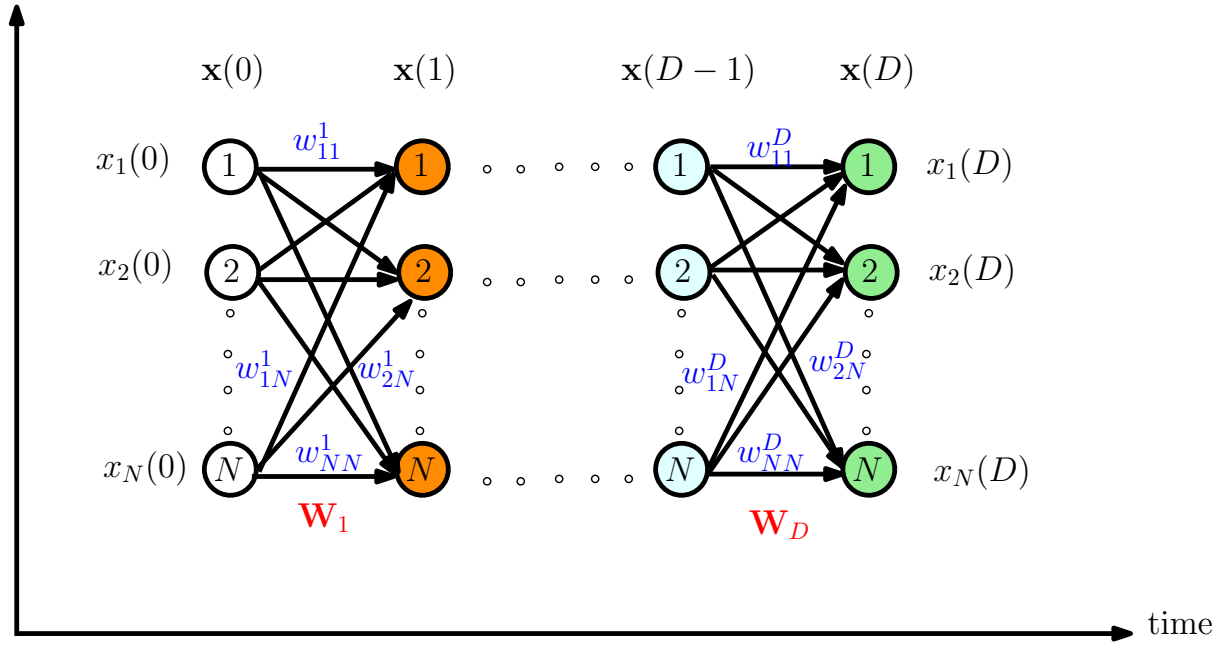


Figure 3.2: Linear iteration scheme in space and time.

Remark 6

The weight matrix $\mathbf{W}(k) \in \mathcal{S}_G$ is associated with the graph G. Therefore, the entries $w_{ij}(k)$ are designed following the rule defined as:

$$w_{ij}(k) = \begin{cases} w_{ij}(k), & \text{if } j \in N_i \\ 0, & \text{otherwise} \end{cases}$$

To simplify the notation, we denote $\mathbf{W}(k)$ as $\mathbf{W}_k$.

The linear iteration scheme in space and time can be viewed as a multilayer neural network. The selection of the weights can be analyzed through the scope of training. By using a set of P learning sequences, the mechanism is divided into two main steps, namely, forward step and backward step. *The main idea is to compute the partial derivatives of the error between the actual output and desired output, then propagate them back to each layer of the network in order to update the weights.* The aim is to reduce the error criteria E efficiently.

Remark 7

Indeed, in most systems where communication is involved, learning sequences are used for communication channel identification or for synchronization. These sequences are used during the mandatory configuration step before transmitting informative data, i.e. running average consensus in our case. We assume that all the agents know the

average of the learning sequence. For instance, after running a standard consensus protocol (Algorithm 1) with the weight consensus matrix as mentioned in Subsection 1.2.4 or the minimal time consensus algorithm proposed in [78].

Let $\{x_{i,p}(0), y_{i,p}\}$, $i = 1, \cdots, N$, $p = 1, \cdots, P$, be the input-output pairs defining the learning sequences, with $y_{i,p} = y_p = \frac{1}{N}\sum_{i=1}^{N} x_{i,p}(0)$. Our aim is to estimate the factor matrices $\mathbf{W}_k$, $k = 1, \cdots, D$, by minimizing the quadratic error

$$E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\sum_{i=1}^{N}\sum_{p=1}^{P}(x_{i,p}(D) - y_p)^2, \tag{3.3.1}$$

with $x_{i,p}(k) = \sum_{j \in N_i \cup \{i\}} w_{ij}(k) x_{j,p}(k-1)$, $w_{ij}(k)$ being the entries of matrices $\mathbf{W}_k \in \mathcal{S}_G$. We can rewrite the cost function (3.3.1) as

$$E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\left\| \prod_{k=D}^{1} \mathbf{W}_k \mathbf{X}(0) - \mathbf{Y} \right\|_F^2, \; s.t \; \mathbf{W}_k \in \mathcal{S}_G, \tag{3.3.2}$$

where $\|.\|_F$ stands for the Frobenius norm, $\mathbf{Y} = \mathbf{J}_N \mathbf{X}(0)$, $\mathbf{Y}$ and $\mathbf{X}(0)$ being $N \times P$ matrices with $y_{i,p}$ and $x_{i,p}$ as entries, respectively.

We assume that $\mathbf{X}(0)\mathbf{X}(0)^T = \mathbf{I}_N$ which means that the input vector is orthogonal. For instance, vectors of the canonical basis of $\mathbb{R}^N$ can be used as inputs. Hence, we can note that $E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\left\| \prod_{k=D}^{1} \mathbf{W}_k - \mathbf{J}_N \right\|_F^2$, meaning that minimizing (3.3.2) is equivalent to solving the factorization problem (3.2.5):

$$\{\mathbf{W}_k^*\}|_{k=1,\ldots,D} = \arg \min_{\{\mathbf{W}_k\}_{t=1,\ldots,D}} \frac{1}{2}\sum_{p=1}^{P} tr\left(\epsilon_p(\mathbf{W})\epsilon_p^T(\mathbf{W})\right), \tag{3.3.3}$$

with

$$\epsilon_p(\mathbf{W}) = \prod_{k=D}^{1} \mathbf{W}_k \mathbf{x}_p(0) - \mathbf{y}_p, \tag{3.3.4}$$

where $tr(.)$ denotes the trace operator and $\mathbf{y}_p = \mathbf{J}_N \mathbf{x}_p(0)$.

Denoting $E_p(\mathbf{W}) = \frac{1}{2} tr\left(\epsilon_p(\mathbf{W})\epsilon_p^T(\mathbf{W})\right)$, the solution of this optimization problem can then be obtained iteratively by means of a gradient descent method:

$$\mathbf{W}_k := \mathbf{W}_k - \alpha \sum_{p=1}^{P} \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_k}$$

or a stochastic gradient one:

$$\mathbf{W}_k := \mathbf{W}_k - \alpha \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_k},$$

where the gradient of the cost function is approximated by the gradient at a single input-output sequence. In what follows, we resort to a stochastic gradient method. For this purpose, we first state the following technical lemma:

Lemma 3

The derivatives of the cost function $E_p(\boldsymbol{W}) = \frac{1}{2} tr\left(\epsilon_p(\boldsymbol{W})\epsilon_p^T(\boldsymbol{W})\right)$ *with* $\epsilon_p(\boldsymbol{W})$ *defined in (3.3.4) can be computed as follows:*

$$\frac{\partial E_p(\boldsymbol{W})}{\partial \boldsymbol{W}_D} = \boldsymbol{\delta}_{D,p}\mathbf{x}_p^T(D-1) \tag{3.3.5}$$

$$\frac{\partial E_p(\boldsymbol{W})}{\partial \boldsymbol{W}_k} = \boldsymbol{\delta}_{k,p}\mathbf{x}_p^T(k-1), k = 1, \ldots, D-1, \tag{3.3.6}$$

where $\boldsymbol{\delta}_{D,p} = \mathbf{x}_p(D) - \bar{\mathbf{x}}_p$ *is the difference between the actual output and the desired output with* $\bar{\mathbf{x}}_p = \mathbf{y_p} = \frac{1}{N}\sum_{i=1}^{N} x_{i,p}(0)$*; and*

$$\boldsymbol{\delta}_{k-1,p} = \boldsymbol{W}_k^T \boldsymbol{\delta}_{k,p}, k = 1, \ldots, D. \tag{3.3.7}$$

Proof: The consensus network being a linear system we know that:

$$\mathbf{x}_p(k) = \mathbf{W}_k \mathbf{x}_p(k-1),$$

Therefore, we can explicitly write the output according to the weighting matrix of interest, i.e.

$$\begin{aligned} \mathbf{x}_p(D) &= \mathbf{W}_D \mathbf{x}_p(D-1) \\ &= \prod_{j=D}^{k+1} \mathbf{W}_j \mathbf{W}_k \mathbf{x}_p(k-1), k = 1, \cdots, D-1. \end{aligned}$$

Equivalently, by defining $\mathbf{Z}_{k+1} = \prod_{j=D}^{k+1} \mathbf{W}_j$, we get $\mathbf{x}_p(D) = \mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1)$. The cost function can be written as

$$E_p(\mathbf{W}) = \frac{1}{2} tr((\mathbf{W}_D\mathbf{x}_p(D-1) - \bar{\mathbf{x}}_p)(\mathbf{W}_D\mathbf{x}_p(D-1) - \bar{\mathbf{x}}_p)^T).$$

We can easily deduce that $\frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_D} = \boldsymbol{\delta}_{D,p}\mathbf{x}_p^T(D-1)$ with $\boldsymbol{\delta}_{D,p} = \mathbf{x}_p(D) - \bar{\mathbf{x}}_p$.

Now, we can express the cost function according to any matrix $\mathbf{W}_t$, $t = 1, \cdots, D-1$ as

$$E_p(\mathbf{W}) = \frac{1}{2} tr((\mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1) - \bar{\mathbf{x}}_p)(\mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1) - \bar{\mathbf{x}}_p)^T).$$

Expanding the above expression and taking into account the linearity of the trace operator yield

$$\begin{aligned} E_p(\mathbf{W}) = \frac{1}{2}[&tr(\mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1)\mathbf{x}_p(k-1)^T\mathbf{W}_k^T\mathbf{Z}_{k+1}^T) \\ &- tr(\mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1)\bar{\mathbf{x}}_p^T) \\ &- tr(\bar{\mathbf{x}}_p\mathbf{x}_p(k-1)^T\mathbf{W}_k^T\mathbf{Z}_{k+1}^T) \\ &- tr(\bar{\mathbf{x}}_p\bar{\mathbf{x}}_p^T)]. \end{aligned}$$

Now, computing the derivative, we get:

$$\begin{aligned} \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_k} &= \frac{1}{2}[2 \times \mathbf{Z}_{k+1}^T\mathbf{Z}_{k+1}\mathbf{W}_k\mathbf{x}_p(k-1)\mathbf{x}_p(k-1)^T \\ &\quad - 2 \times \mathbf{Z}_{k+1}^T\bar{\mathbf{x}}_p\mathbf{x}_p(k-1)^T] \\ &= \mathbf{Z}_{k+1}^T(\mathbf{Z}_{k+1}\mathbf{W}_t\mathbf{x}_p(k-1) - \bar{\mathbf{x}})\mathbf{x}_p(k-1)^T \\ &= \mathbf{Z}_{k+1}^T\underbrace{(\mathbf{x}_p(D) - \bar{\mathbf{x}}_p)}_{\boldsymbol{\delta}_{D,p}}\mathbf{x}_p(k-1)^T \\ &= \mathbf{Z}_{k+1}^T\boldsymbol{\delta}_{D,p}\mathbf{x}_p(k-1)^T \\ &= \mathbf{W}_{k+1}^T\mathbf{W}_{k+2}^T\dots\underbrace{\mathbf{W}_D^T\boldsymbol{\delta}_D}_{\boldsymbol{\delta}_{D-1,p}}\mathbf{x}_p(k-1)^T \\ &= \mathbf{W}_{k+1}^T\boldsymbol{\delta}_{k+1,p}\mathbf{x}_p(k-1)^T \\ &= \boldsymbol{\delta}_{k,p}\mathbf{x}(k-1)^T. \end{aligned}$$

■

Applying the results of Lemma 3, the updating scheme of the optimization algorithm is as follows:

$$\mathbf{W}_k[t+1] = \mathbf{W}_k[t] - \alpha\frac{\partial E_{p(t)}(\mathbf{W})}{\partial \mathbf{W}_k} = \mathbf{W}_k[t] - \alpha\boldsymbol{\delta}_{k,p(t)}\mathbf{x}_{p(t)}^T(k-1), \tag{3.3.8}$$

where $p(t) \in \{1, 2, \cdots, P\}$, and t stands for the t^{th} iteration of the optimization process. Then, we project the updated weights on $\mathcal{S}_G$ that gives:

$$\begin{aligned} \mathbf{W}_k[t+1] &= \mathbf{W}_k[t] - \alpha\frac{\partial E_{p(t)}(\mathbf{W})}{\partial \mathbf{W}_k} \\ &= \mathbf{W}_k[t] - \alpha\left(\left[\boldsymbol{\delta}_{k,p(t)}\mathbf{x}_{p(t)}^T(k-1)\right] \circ (\mathbf{I}_N + \mathbf{A})\right), \end{aligned} \tag{3.3.9}$$

The gradient descent algorithm (3.3.9) acts by alternating the following steps: the learning sequence is first propagated forward. Then the error between the targeted output and $\mathbf{x}(D)$ is computed and then propagated .

Proposition 1

Let $(\boldsymbol{W}_1^*, \dots, \boldsymbol{W}_D^*)$ be a local minimum, of $E(\boldsymbol{W}_1, \dots, \boldsymbol{W}_D)$, then

$$\prod_{k=D}^{1} \boldsymbol{W}_k^* = \boldsymbol{J}_N$$

.

Proof: Let $\mathbf{W}_1^*, \ldots, \mathbf{W}_D^*$ be the local minimum of the problem (3.3.2), then the derivatives

$$\frac{\partial E_p(\mathbf{W}^*)}{\partial \mathbf{W}_k^*}|_{k=1,\ldots,D} = 0.$$

Following Lemma 3,

$$\boldsymbol{\delta}_{k,p}\mathbf{x}_p^T(k-1) \quad = \quad 0, \quad k = 1, \ldots, D.$$

Thus,

$$\begin{aligned} & \boldsymbol{\delta}_{D,p}\mathbf{x}_p^T(D-1) && = 0 \\ \Leftrightarrow \quad & \boldsymbol{\delta}_{D,p} && = 0 \\ \Leftrightarrow \quad & \mathbf{x}_p(D) && = \bar{\mathbf{x}}_p, \quad p = 1, \ldots, P. \end{aligned}$$

It can be rewritten in matrix form:

$$\begin{aligned} & \prod_{k=D}^{1} \mathbf{W}_k^* \mathbf{X}(0) && = \mathbf{J}_N \mathbf{X}(0) \\ \Leftrightarrow \quad & \left(\prod_{k=D}^{1} \mathbf{W}_k^* - \mathbf{J}_N \right) \mathbf{X}(0) && = 0. \end{aligned}$$

Since $\mathbf{X}(0)\mathbf{X}^T(0) = \mathbf{I}_N$, then

$$\begin{aligned} & \prod_{k=D}^{1} \mathbf{W}_k^* - \mathbf{J}_N && = 0 \\ \Leftrightarrow \quad & \prod_{k=D}^{1} \mathbf{W}_k^* && = \mathbf{J}_N. \end{aligned}$$

■

This mechanism is similar to the gradient back-propagation algorithm. Its convergence has been well studied in the literature, see [14] and [50, 73] for instance. Convergence towards a local minimum is guaranteed if the step-size α is appropriately chosen inside the interval $(0, 1)$. If the step-size α is set too large, the error E tends to oscillate, never reaching a minimum. In contrast, too small α prevents the optimization from making reasonable progress. Therefore, choosing an appropriate α is also an encountered difficulty in practice. For a particular problem, choosing a suitable stepsize α involves experimenting with different values to see how the convergence reacts. Such a step-size also influences on speed of convergence of the consensus algorithm. Several rules have been proposed for accelerating the convergence speed such as using the adaptive learning rate α [29] or adaptive momentum [69] . However, these techniques are not consistent

with the distributed framework of the proposed algorithm. One trick to speed-up the convergence is to add a regularization term in the cost function to be minimized.

$$\{\mathbf{W}_k^*\}|_{k=1,\dots,D} = \arg \min_{\{\mathbf{W}_k\}_{k=1,\dots,D}} \sum_{p=1}^{P} E_{p(t)}(\mathbf{W}) + \frac{1}{2}\sum_{k=1}^{D} \beta \|\mathbf{W}_k[t] - \mathbf{W}_k[t-1]\|^2.$$

By minimizing such a cost function, the update equation is given by:

$$\mathbf{W}_k[t+1] = \mathbf{W}_k[t] - \alpha \left(\left[\boldsymbol{\delta}_{k,p(t)} \mathbf{x}_{p(t)}^T(k-1) \right] \circ (\mathbf{I}_N + \mathbf{A}) \right) + \beta(\mathbf{W}_k[t] - \mathbf{W}_k[t-1]).$$

Entry-wise, each agent updates its entries as follows:

$$w_{ij}^k[t+1] := w_{ij}^k[t] - \alpha[t](\delta_{i,k} x_j(k-1)\hat{a}_{ij}) + \beta(w_{ij}^k[t] - w_{ij}^{(k-1)}[t-1]),$$

where:

- $\delta_{i,k}$ is the i-th entry of $\boldsymbol{\delta}_k$ and $x_j(k-1)$ the j-th entry of $\mathbf{x}(k-1)$. Here, $\hat{a}_{ij}$-the entry of matrix $(\mathbf{I}_N + \mathbf{A})$ is defined as:

$$\hat{a}_{ij} = \begin{cases} a_{ij} & \text{if } i \neq j. \\ 1 & \text{if } i = j. \end{cases}$$

- α and β are the learning rate and the momentum rate respectively. They can be chosen as constants or as time varying parameters. In our study, the learning rate $\alpha[t]$ varies at each optimization iteration step t while the momentum rate is fixed.

Remark 8

In order to avoid the confusion between the notation of the time-step k of the finite-time average consensus protocol and the optimization iteration t, we denote $w_{ij}(k)$ as w_{ij}^k.

For a clear understanding, Figure 3.3 shows the mechanism of the proposed algorithm. In detail, for each learning sequence, the algorithm first executes the forward step, where each node updates repeatedly its value. At the end of this step, each layer has stored the updated values, then the partial derivatives of the error δ_D between $\mathbf{x}(D)$ and the desired average consensus values $\bar{\mathbf{x}}$ are estimated. The backward step starts by propagating back the estimated partial derivatives δ_D to each layer until the input layer receives δ_1. Now, all weights can be updated. The regime continues with the next learning sequence. The mechanism is repeated until the error criterion E satisfies a given threshold θ, which is a significantly small number.

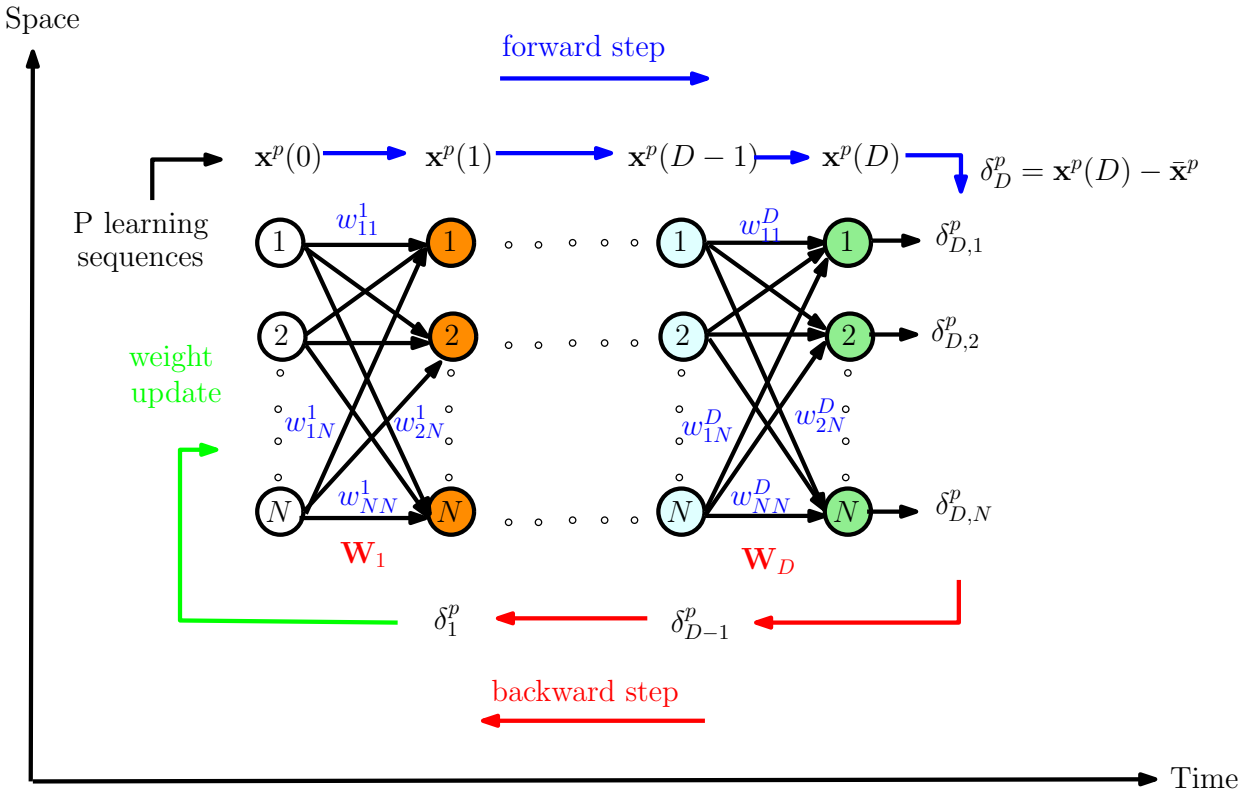


Figure 3.3: Mechanism of the proposed back-propagation based method.

The distributed algorithm is then described as follows:

Algorithm 3

1. *Initialization:*
   - *Number of steps* D*, number of patterns* P
   - *Learning sequence* $\{x_{i,p}(0), y_p\}$*, where* $i = 1, \cdots, N, \quad p = 1, \cdots, P$*, with* $y_p = \frac{1}{N}\sum_{i=1}^{N} x_{i,p}(0)$.
   - *Random initial weight matrices* $\boldsymbol{W}_k[0], k = 1, \ldots, D$*, and* $\boldsymbol{W}_k[-1] = \boldsymbol{0}$
   - *Learning rate:* $0 < \alpha[0] < 1$;
   - *Momentum terms:* $0 < \beta < 1$;
   - *Select a threshold* θ
   - *Set* $t = 0$
2. *Set* $p = 0$;
   (a) *Set* $p := p + 1$,

(b) *Select the corresponding input-output sequence* $x_i(0) = x_{i,p}(0)$, $\bar{x} = y_p$.

(c) *Learning sequence propagation:*

$$x_i(k) = \sum_{j \in N_i \cup \{i\}} w_{ij}^k[t] x_j(k-1), \quad k = 1, \cdots, D.$$

(d) *Error computation:*

$$\delta_{i,D} = x_i(D) - \bar{x}; \quad e_{i,p} = \delta_{i,D}^2.$$

(e) *Error propagation:*

$$\delta_{i,k-1} = \sum_{j \in N_i \cup \{i\}} w_{ji}^k[t] \delta_{j,k}, \quad k = D, \cdots, 2.$$

(f) *Matrices updating: for* $k = 1, \cdots, D$, $i = 1, \cdots, N$, *and* $j \in N_i \cup \{i\}$,

$$w_{ij}^k[t+1] \;=\; w_{ij}^k[t] - \alpha[t]\delta_{i,k} x_j(k-1)\hat{a}_{ij} + \beta(w_{ij}^k[t] - w_{ij}^k[t-1]).$$

(g) *Increment* t.

(h) *If* $p = P$, *compute the mean square error* $E_i = \frac{1}{N} \sum_{p=1}^{P} e_{i,p}$, *else return to 2a.*

(i) *If* $E_i < \theta$ *stop the learning process, else return to 2.*

3.4 Numerical results

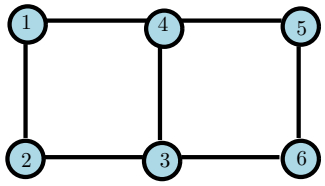


Figure 3.4: 6-node graph

In this section, we consider two examples of different network topologies to evaluate the developed algorithm. The learning sequence is constituted by the vectors of the canonical basis of $\mathbb{R}^N$. The performance of designed protocols is evaluated by means of the mean

square error (Mean Square Error (MSE)):

$$MSE = \frac{1}{NP}\sum_{i=1}^{N}\sum_{p=1}^{P}(x_{i,p}(D) - y_p)^2.$$

3.4.1 Example 1

Let us consider the graph depicted in Figure 3.4. Its diameter is $d(G) = 3$, hence the bounds of the number of iterations D are $d(G) < D < N - 1 = 5$. The Laplacian matrix is defined as follows:

$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

The eigenvalues of the Laplacian matrix $\mathbf{L}$ are: 0, 1, 2, 3 and 5. According to [42], the factorization of the average consensus matrix by means of graph Laplacian-based consensus matrices needs D factor matrices, D being the number of nonzero distinct eigenvalues of the Laplacian matrix, this number being, in general, greater than or equal to the diameter of the graph. Therefore, we get the following factorization:

$$(\mathbf{I}_N - \mathbf{L})(\mathbf{I}_N - \frac{1}{2}\mathbf{L})(\mathbf{I}_N - \frac{1}{3}\mathbf{L})(\mathbf{I}_N - \frac{1}{5}\mathbf{L}) = \mathbf{J}_6,$$

meaning that consensus is achieved in 4 steps.

$$\mathbf{J}_6 = \begin{pmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{pmatrix}$$

Since the optimal number of steps is not a priori known, we run the algorithm for different values of D taken in the interval of $[d(G), N - 1]$. Here, the possible values of D are $\{3, 4, 5\}$. Figures 3.5 shows that the best solution is obtained for $D = 3$. With the proposed algorithm we get a number of step lower than that given by Laplacian based

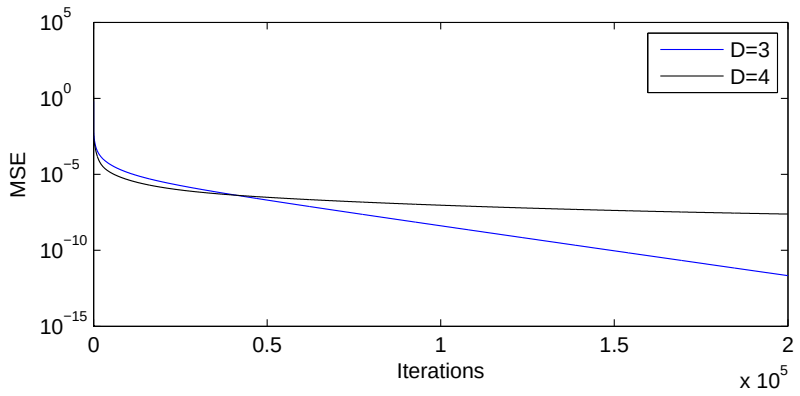


Figure 3.5: MSE comparison for the two possible numbers of factors

matrices proposed in [41, 42]. The weight matrices are:

$$\mathbf{W}_1 = \begin{pmatrix} -0.4028 & 0.0415 & 0 & 0 & 0 & 0 \\ 0.4418 & -0.0455 & 0 & 0 & 0 & 0 \\ 0.5749 & 0 & 0.0537 & 0.0537 & 0.9219 & 0 \\ 0 & 0.4053 & 0.3675 & 0.3675 & 0 & 0.3902 \\ 0 & 0 & 0 & 0 & -1.3034 & 0.0806 \\ 0 & 0 & 0 & 0 & 0.1122 & -0.0069 \end{pmatrix}$$

$$\mathbf{W}_2 = \begin{pmatrix} 0.7767 & 0.2508 & 0.1706 & 0 & 0 & 0 \\ 0.0201 & 0.6826 & 0 & 0.4527 & 0 & 0 \\ 0.6941 & 0 & 0.7680 & 0.2985 & 0.4385 & 0 \\ 0 & 0.2560 & 0.1578 & 0.5424 & 0 & 0.6013 \\ 0 & 0 & 0.3756 & 0 & 0.5232 & 0.8876 \\ 0 & 0 & 0 & 0.2684 & -0.1270 & 0.5548 \end{pmatrix}$$

$$\mathbf{W}_3 = \begin{pmatrix} 0.7429 & 0.6296 & 0.3653 & 0 & 0 & 0 \\ 0.2881 & 0.2741 & 0 & 0.5699 & 0 & 0 \\ 0.7055 & 0 & 0.6403 & 0.2695 & 0.3771 & 0 \\ 0 & 0.4246 & 0.0716 & 0.1496 & 0 & 0.5491 \\ 0 & 0 & 0.4218 & 0 & 0.4556 & 0.9515 \\ 0 & 0 & 0 & 0.4958 & 0.3039 & 0.5832 \end{pmatrix}$$

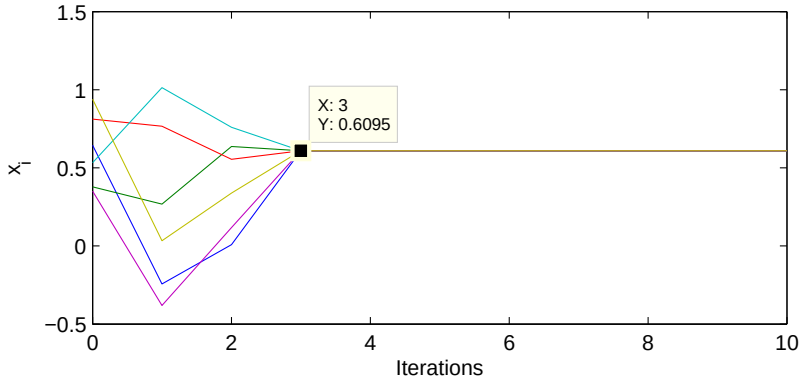
We can easily check that:

$$\mathbf{W}_3\mathbf{W}_2\mathbf{W}_1 = \mathbf{J}_6.$$

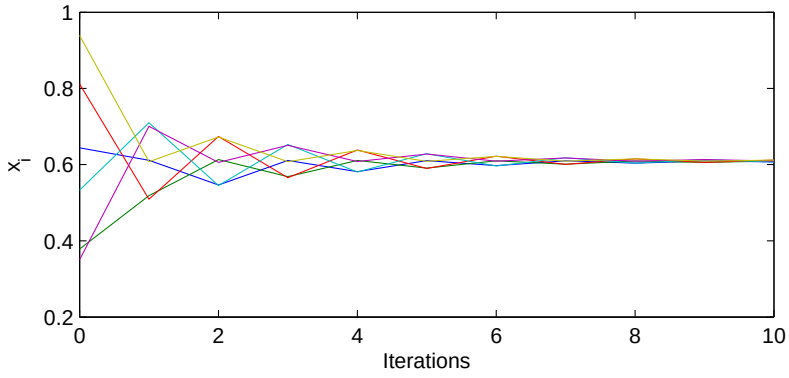
Now, let us consider the finite-time consensus protocol in its execution step, i.e. after the configuration step. For arbitrary initial values, the trajectories of the state of the network are depicted in Figure 3.6(a). Exact average consensus is achieved in 3 steps. When using the optimal constant edge weights protocol $\mathbf{W} = \mathbf{I}_N - \alpha\mathbf{L}$ where $\alpha = \frac{2}{\lambda_2(\mathbf{L})+\lambda_N(\mathbf{L})}$ proposed in [74], we get the trajectories in Figure 3.6(b). Showing that more iterations

are actually needed to reach consensus.

$$\mathbf{W}_{\alpha=0.3333} = \begin{pmatrix} 0.3333 & 0.3333 & 0 & 0.3333 & 0 & 0 \\ 0.3333 & 0.3333 & 0.3333 & 0 & 0 & 0 \\ 0 & 0.3333 & 0 & 0.3333 & 0 & 0.3333 \\ 0.3333 & 0 & 0.3333 & 0 & 0.3333 & 0 \\ 0 & 0 & 0 & 0.3333 & 0.3333 & 0.3333 \\ 0 & 0 & 0.3333 & 0 & 0.3333 & 0.3333 \end{pmatrix}$$



(a)



(b)

Figure 3.6: (a)Trajectory of the proposed finite-time consensus protocol
(b)Trajectory of an asymptotic consensus protocol by using optimal constant edge weights

3.4.2 Example 2

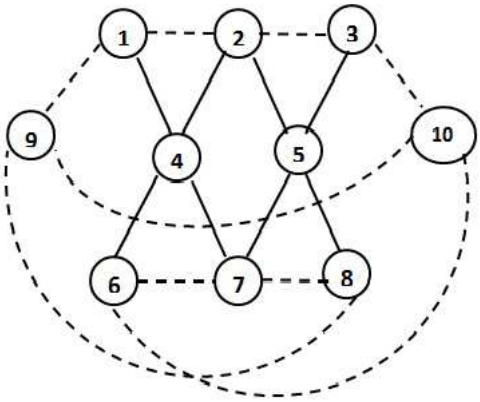


Figure 3.7: 10-node graph

Now considering the network in Figure 3.7, we can see that diameter $d(G)$ of this graph is equal to 2, and the radius is $r(G) = 2$.

The Laplacian matrix is defined as:

$$\mathbf{L} = \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 3 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 4 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 4 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 3 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 3 \end{pmatrix}$$

The nonzero corresponding eigenvalues are $\{2, 5, 6\}$. Hence, if the Laplacian spectrum is known, then we can find the consensus matrices based on the Laplacian spectrum. That are: $\mathbf{W}_k = \mathbf{I}_N - \frac{1}{\lambda_k}\mathbf{L}, k = 1, 2, 3$. Therefore,

$$(\mathbf{I}_N - \frac{1}{2}\mathbf{L})(\mathbf{I}_N - \frac{1}{5}\mathbf{L})(\mathbf{I}_N - \frac{1}{6}\mathbf{L}) = \mathbf{J}_{10}.$$

Running the proposed algorithm, the performance in Figure 3.8 tells us that this network cannot achieve average consensus in 2 steps, which is the diameter of a graph as suggested

in [30]. Such a result has been demonstrated analytically by [34]. However, by running the algorithm as well as varying the value of D in the interval $[d(G), 2r(G)]$, we can pick the optimal D in this case. The final MSE was obtained by considering a sufficient large number of iterations.

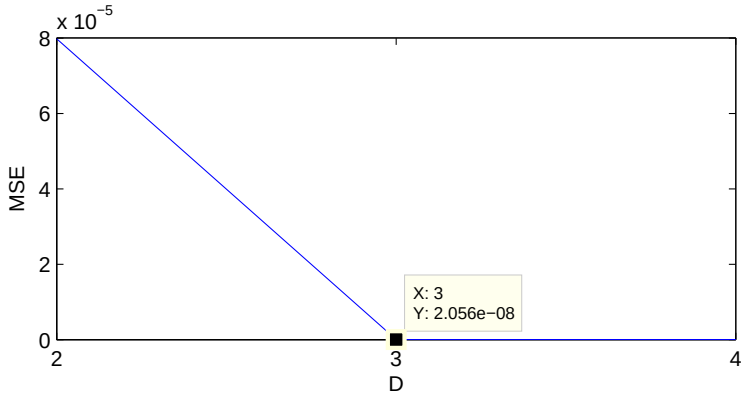


Figure 3.8: Final MSE comparison for different values of D

For this topology, $D = 3$ gives us the best solution for finite-time average consensus, see Figure 3.8. Also, the trajectory of an arbitrary state vector is depicted in Figure 3.9 after getting the sequence of weight matrices.

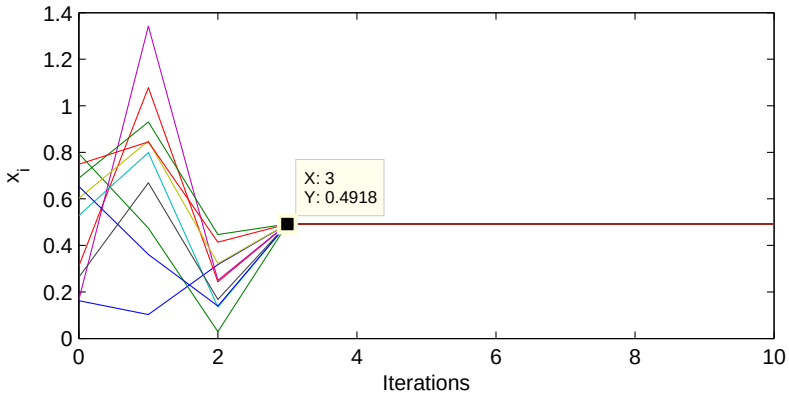


Figure 3.9: Trajectory of an arbitrary state vector for a 10-node network

The sequence of consensus matrices are:

$$\mathbf{W}_1 = \begin{pmatrix} 0.0982 & 0.1094 & 0 & 0.0038 & 0 & 0 & 0 & 0 & -0.0027 & 0 \\ 0.4403 & 0.5004 & 0.0284 & 0.0069 & -0.0448 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3763 & 0.8961 & 0 & 0.6795 & 0 & 0 & 0 & 0 & 0.5185 \\ 0.5459 & 0.5896 & 0 & 0.1849 & 0 & 0.1786 & 0.1404 & 0 & 0 & 0 \\ 0 & 0.5217 & 0.5698 & 0 & 0.1776 & 0 & 0.6942 & 0.8230 & 0 & 0 \\ 0 & 0 & 0 & 0.6030 & 0 & 0.6134 & 0.6207 & 0 & 0 & -0.0039 \\ 0 & 0 & 0 & 0.3971 & 0.1699 & 0.4048 & 0.4035 & 0.1239 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3485 & 0 & 0.0565 & 0.3863 & 0.0524 & 0 \\ 0.2731 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4677 & 0.5781 & 0.2431 \\ 0 & 0 & 0.5508 & 0 & 0 & 0.0033 & 0 & 0 & 0.4278 & 0.5038 \end{pmatrix}$$

$$\mathbf{W}_2 = \begin{pmatrix} -0.1600 & -0.4346 & 0 & 0.5478 & 0 & 0 & 0 & 0 & 0.1101 & 0 \\ 0.5663 & 0.0180 & 0.1650 & -0.3594 & 0.0534 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.4852 & -0.1868 & 0 & -0.0971 & 0 & 0 & 0 & 0 & 0.4086 \\ 0.5540 & 0.2436 & 0 & -0.2767 & 0 & -0.1164 & 0.4223 & 0 & 0 & 0 \\ 0 & 0.2732 & -0.0974 & 0 & -0.0435 & 0 & 0.3421 & 0.1526 & 0 & 0 \\ 0 & 0 & 0 & 0.1143 & 0 & -0.1298 & 0.0912 & 0 & 0 & 0.3312 \\ 0 & 0 & 0 & 0.0910 & -0.0267 & 0.3045 & -0.3423 & 0.2817 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0127 & 0 & 0.1642 & 0.2147 & -0.0695 & 0 \\ 0.6788 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2566 & -0.0812 & 0.4254 \\ 0 & 0 & 0.2218 & 0 & 0 & 0.3494 & 0 & 0 & 0.3438 & -0.5230 \end{pmatrix}$$

$$\mathbf{W}_3 = \begin{pmatrix} 0.7934 & 0.1399 & 0 & 0.6098 & 0 & 0 & 0 & 0 & 0.3429 & 0 \\ 0.9671 & 0.8649 & 0.2048 & -0.0068 & 0.4405 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0336 & 0.6163 & 0 & 0.5474 & 0 & 0 & 0 & 0 & 0.4221 \\ 0.9109 & 0.4391 & 0 & 0.0472 & 0 & 0.2317 & 0.6459 & 0 & 0 & 0 \\ 0 & 0.6592 & 0.6413 & 0 & 0.2079 & 0 & 0.8466 & 0.8736 & 0 & 0 \\ 0 & 0 & 0 & 0.7624 & 0 & 0.6332 & 0.9063 & 0 & 0 & 0.0789 \\ 0 & 0 & 0 & 0.7259 & 0.2381 & 0.6906 & 0.4937 & 0.2066 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5763 & 0 & 0.0855 & 0.2050 & 0.6821 & 0 \\ 0.8189 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5641 & 0.4207 & -0.0849 \\ 0 & 0 & 0.1202 & 0 & 0 & 0.1029 & 0 & 0 & 0.5202 & 0.4767 \end{pmatrix}$$

As the result, the product of consensus matrices gives out the averaging matrix.

$$\mathbf{W}_3\mathbf{W}_2\mathbf{W}_1 = \mathbf{J}_{10} = \frac{1}{10}\mathbf{1}\mathbf{1}^T.$$

3.5 Conclusion and discussion

In this chapter, we have proposed a way for distributively designing the finite-time average consensus protocol. By using a learning sequence, we have shown how to solve the matrix factorization problem (3.2.5) in a fully distributed way. The method is based on the gradient back-propagation method. The factorization gives rise to factor matrices that are not necessarily symmetric or stochastic. Given the diameter of the graph $d(G)$, we can find out the optimal value of the number of steps necessary for reaching average

consensus by varying the value of D in the interval $(d(G) \leq D \leq N-1)$. However, it is obviously inconvenient for large-scale networks.

On the other hand, we can see that, the obtained consensus matrices are not doubly stochastic matrices.

Finally, the speed of convergence of the proposed algorithm is still an open issue. For this purpose, future works encompass optimal step-size for the gradient descent based method and other optimization methods with a particular focus on large size graphs. In addition, the dependency of the convergence speed on the learning sequence is to be studied. Hence, another interesting issue will be the design of optimal learning sequences. Beside that, the robustness of the finite-time average consensus should be definitely paid a great attention.

Chapter 4

Distributed network robustness assessment

Contents

4.1 Introduction

The efficiency of a network is evaluated through its functionality and its robustness. According to the latter property, some questions may arise. For instance, if there is an accidental event, how does the network react? Will the network survive? In order to answer these questions, the study of the robustness of networks has increasingly attracted the attention of the scientific community.

In the literature, there is quite a lot of robustness measures (see Section 1.3), [20, 21]. For instance, the earliest measure is based on the connectivity of the graph representing

the network [16]. In the other words, a network is robust if there exists the presence of alternate paths, which ensure the possibility of communication in spite of damage [16]. There are two classical concepts of connectivity for a graph which can be used to model network robustness that are the vertex and edge connectivities. The vertex (edge) connectivity $\mathcal{K}_v(\mathcal{K}_e)$ is the minimal number of vertices (edges) to be removed in order to disconnect the given graph (see Subsection 1.3.1). By this analytical approach, in order to deduce the robustness index, we have to remove each edge to check for the connectivity of the graph. Even if this approach is simple, it is also a burden if we analyse reasonable larger complex networks.

In [72], the authors have proposed the concept of natural connectivity as a spectral measure of robustness in complex networks. The natural connectivity is expressed in mathematical form as the average eigenvalue of the adjacency matrix of the graph representing the network topology. The advantage of this method is that the proposed measure works in both connected and disconnected networks. On the other hand, the Laplacian spectrum $sp(\mathbf{L})$ can be used to compute the robustness indices that evaluate the performance of the given network. For instance, the second smallest eigenvalue of the Laplacian spectrum $\lambda_2(\mathbf{L})$, which is also known as algebraic connectivity [28], is well-known as a critical parameter that influences on the performance and robustness of dynamical systems due to its main role in the connectivity of the graph [20]. Hence, there is a lot of researches on this algebraic connectivity in the literature [3, 51, 76]. Furthermore, there are other remarkable measures for getting the desired robustness indices: the effective graph resistance $\mathcal{R}$ and the number of spanning tree ξ (see Section 1.3), which are based on the whole non-zero Laplacian spectrum.

We propose distributed methods to estimate both the effective graph resistance $\mathcal{R}$ and the number of spanning trees ξ. Figure 4.1 shows that the distributed estimation of the network robustness is divided into two main stages including Laplacian spectrum $sp(\mathbf{L})$ estimation, effective graph resistance and number of spanning trees $(\mathcal{R}, \xi)$ computation. Firstly, we propose consensus-based methods to estimate the whole spectrum of the Laplacian matrix in a distributed way. Primarily, we compute the average consensus value $\bar{x}$ by using the standard consensus protocols (Algorithm 1) or the decentralized minimum-time consensus value computation algorithm [78]. Next, the estimation of the Laplacian spectrum can be divided in two steps that are the estimation of the distinct Laplacian eigenvalues, and then evaluation of the corresponding multiplicities. Finally, from the estimated $sp(\mathbf{L})$ the robustness indices can be computed. Our focus is therefore on the estimation of the Laplacian spectrum $sp(\mathbf{L})$.

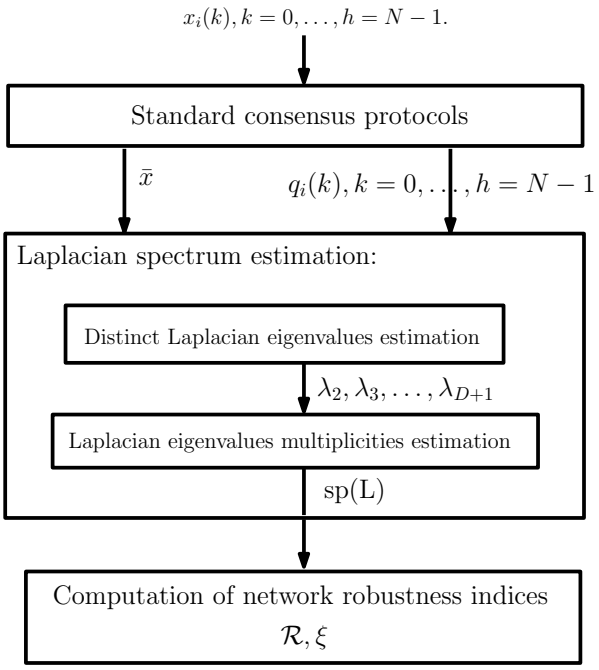


Figure 4.1: Diagram of distributed estimation of network robustness.

4.2 Literature review

The estimation of the Laplacian spectrum $sp(\mathbf{L})$ can be executed by means of centralized algorithms where a global knowledge about the network topology is available. However, in order to circumvent the inherent vulnerability of centralized schemes, various studies have been carried out on decentralized algorithms for estimating the Laplacian eigenvalues. Most of them are restricted to the estimation of the second smallest Laplacian eigenvalue $\lambda_2(\mathbf{L})$. For instance, in [77], the second smallest Laplacian eigenvalue was estimated by resorting to a continuous-time decentralized power iteration method that enables for each node i to estimate x_i, where $\mathbf{x} = [x_1 \ \ldots \ x_N]^T$ being the estimate of the eigenvector $\mathbf{v}_2$, corresponding to $\lambda_2(\mathbf{L})$. In [3], the power iteration method consists of transforming the Laplacian matrix $\mathbf{L}$ associated with the undirected and connected graph G into a new matrix $\mathbf{C}$, which is the deflated version of the Perron matrix of $\mathbf{L}$, where the algebraic connectivity $\lambda_2(\mathbf{L})$ becomes the leading eigenvalue, i.e. the spectral radius $\rho(\mathbf{C})$ is an expression that depends on $\lambda_2(\mathbf{L})$. First, nodes compute the powers of a deflated Laplacian matrix $\mathbf{C}$, $\mathbf{C}^k$ in a distributed way. Then, at each time step k for each node i, the sum $c_i(k)$ of the absolute values of the entries of $\mathbf{C}^k$ is defined to execute the max-consensus on $c_i(k)$, whose maximum value converges to the spectral radius $\rho(\mathbf{C})$ when k tends to infinity. Once $\rho(\mathbf{C})$ is computed, the estimated algebraic connectivity is finally obtained by reversing the transformation.

For computing the whole Laplacian spectrum, several approaches have been considered in the recent literature: Fast Fourier Transform (FFT) based methods [27, 67], local

eigenvalue decomposition of given observability based matrices [40].

For FFT-based methods, the main idea is to make the state of each agent oscillate only at frequencies corresponding to the eigenvalues of the network topology. The problem is then mapped into a signal processing one that can be efficiently and independently solved by each agent by means of FFT. In [27], the amplitude and phase of the oscillations have been characterized as function of the eigenvectors of the Laplacian and the initial conditions. In this paper, a connected undirected graph $G(V, E)$ with $N = |V|$ has been considered. From an initial condition $x_i(0), z_i(0)$ generated from a uniform distribution in $[-1, 1]$, each node i simulates the following local interaction rule with its neighbors $N_i(t)$:

$$\begin{cases} \dot{x}_i(t) &= z_i(t) + \sum_{j \in N_i}(z_i(t) - z_j(t)), \\ \dot{z}_i(t) &= -x_i(t) - \sum_{j \in N_i}(x_i(t) - x_j(t)), \end{cases}$$

which can be expressed in vector form as:

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{z}}(t) \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{z}(t) \end{bmatrix},$$

where $\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_N - \mathbf{L} \\ \mathbf{I}_N + \mathbf{L} & \mathbf{0} \end{bmatrix}$ is skew symmetric and $\mathbf{0}$ is the null $N \times N$ matrix. To any eigenvalue $\lambda(\mathbf{L})$ it corresponds a couple of complex and conjugate eigenvalues $\lambda(\mathbf{M}), \bar{\lambda}(\mathbf{M})$, that is:

$$\lambda(\mathbf{M}) = j(1 + \lambda(\mathbf{L})), \quad \bar{\lambda}(\mathbf{M}) = -j(1 + \lambda(\mathbf{L})),$$

while the corresponding eigenvectors $v_{\mathbf{M}}$ of $\mathbf{M}$ are function of the eigenvectors $v_{\mathbf{L}}$ of $\mathbf{L}$:

$$v_{\mathbf{M}} = [v_{\mathbf{L}}^T \; j v_{\mathbf{L}}^T]^T, \quad \bar{v}_{\mathbf{M}} = [v_{\mathbf{L}}^T \; -j v_{\mathbf{L}}^T]^T.$$

It follows that each state of each node $x_i(t)$ follows an oscillating trajectory which is a linear combination of sinusoids, whose amplitudes and phases shifted are function of the initial conditions and of the graph topology, oscillating at and only at frequencies corresponding to the eigenvalues of the network at time t. Also, the state trajectory of $x_i(t)$ and $z_i(t)$ can be viewed as a function of Laplacian eigenvectors and the initial conditions:

$$\begin{aligned} x_i(t) &= \sum_{j=1}^{N} [v_{\mathbf{L}j}(i)(cos((1 + \lambda_j(\mathbf{L}))t) v_{\mathbf{L}j}^T \mathbf{x}(0) + sin((1 + \lambda_j(\mathbf{L}))t) v_{\mathbf{L}j}^T \mathbf{z}(0))], \\ z_i(t) &= \sum_{j=1}^{N} [v_{\mathbf{L}j}(i)(-sin((1 + \lambda_j(\mathbf{L}))t) v_{\mathbf{L}j}^T \mathbf{x}(0) + cos((1 + \lambda_j(\mathbf{L}))t) v_{\mathbf{L}j}^T \mathbf{z}(0))]. \end{aligned}$$

In detail, the module of the Fourier transform of the i-th state components $x_i(t)$ and $z_i(t), i = 1, \ldots, N$ can be written as:

$$\begin{aligned} |\mathcal{F}[x_i(t)]| &= |\mathcal{X}_i(f)| = \sum_{j=1}^{m} \frac{a_{j,i}}{2} \delta \left(f \pm \frac{1 + \lambda_j(\mathbf{L})}{2\pi} \right), \\ |\mathcal{F}[z_i(t)]| &= |\mathcal{Z}_i(f)| = \sum_{j=1}^{m} \frac{b_{j,i}}{2} \delta \left(f \pm \frac{1 + \lambda_j(\mathbf{L})}{2\pi} \right), \end{aligned}$$

where f is the frequency domain variable; m is the number of distinct Laplacian eigenvalues; and $a_{j,i}, b_{j,i}$ are coefficients given by:

- For $\lambda_1(\mathbf{L}) = 0$:

$$\begin{aligned} a_{1,i} &= v_1(i) v_1^T \mathbf{x}(0) = \frac{\mathbf{1}_N^T \mathbf{x}(0)}{N}, \\ b_{1,i} &= v_1(i) v_1^T \mathbf{z}(0) = \frac{\mathbf{1}_N^T \mathbf{z}(0)}{N}, \end{aligned}$$

  where v_1 is the unitary norm eigenvector corresponding to $\lambda_1(\mathbf{L}) = 0$.

- For $\lambda_j(\mathbf{L}) > 0$:

$$a_{j,i} = b_{j,i} = \sqrt{\left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} \mathbf{x}(0) \right) \right]^2 + \left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} \mathbf{z}(0) \right) \right]^2},$$

  where $v_j^{(k)}, k = 1, \ldots, \nu_j$ is the unitary norm eigenvector associated to $\lambda_j(\mathbf{L}) > 0$, with ν_j be the algebraic multiplicity.

In a time window of length T, node i estimates the frequencies of the sinusoids of which signal $x_i(t)$ is composed. The frequencies of the spectrum of $x_i(t)$ and $z_i(t)$ are given by $f_j = \pm \frac{1+\lambda_j(\mathbf{L})}{2\pi}$. Hence, $\lambda_j(\mathbf{L}) = 2\pi f_j \pm 1$. However, this approach inherits the limitations of the FFT algorithm. For instance, since $\triangle_f = \frac{f_s}{M}$ where M is the number of samples to be recorded, the resolution of the estimated eigenvalues is strongly dependent on that of the FFT method and the accuracy depends on the amount of stored data (M). On the other hand, the node can only estimate the eigenvalues associated with the mode that is observable. We know that the system is not observable from a single node if there exists an eigenvalue with multiplicity greater than 2. In addition, some nodes may observe only a subset of the eigenvalues. Then, they need to execute additional coordination rules for propagating their data.

In contrast, in [40], an algebraic method using the observability properties of the network was proposed. With this method, the eigenvalues of the network matrix can be recovered by solving a local eigenvalue decomposition problem on an appropriately constructed

matrix of observed data. In this paper, a connected undirected graph $G(V, E)$ with $N = |V|$ has been also considered. Suppose that each node i has some initial values organized in an M-length row vector $\mathbf{x}_i(0), M \geq N$. The dynamics of the network is represented as:

$$\begin{aligned} \mathbf{X}(k+1) &= \mathbf{W}\mathbf{X}(k), \\ \mathbf{Y}_i(k) &= \mathbf{E}_i\mathbf{X}(k), \end{aligned}$$

where:

- $\mathbf{X}(k) \in I\!R^{N\times M}$ is a matrix with $\mathbf{x}_i(k), i = 1, \ldots, N$ as rows;
- $\mathbf{Y}_i(k)$ are the outputs or node values that are seen by node i at the k^{th} time step;
- $\mathbf{E}_i \in I\!R^{\bar{d}_i\times N}$ is the row selection matrix with $\bar{d}_i$ bounded by $[0, d_i + 1]$;
- $\mathbf{W} = \mathbf{I}_N - \epsilon\mathbf{L}$ is the consensus matrix. Its eigenvalue decomposition is $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where the eigenvectors and eigenvalues are respectively organized in the orthogonal matrix $\mathbf{U}$ and the diagonal matrix $\mathbf{D}$.

Since $\mathbf{L} = \mathbf{U}\mathbf{\Delta}\mathbf{U}^T$ with $\mathbf{\Delta}$ being diagonal matrix of $\lambda_i(\mathbf{L})$, then $\mathbf{D} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$.

The purpose of [40] is to estimate the eigenvalues $\lambda_i(\mathbf{L}), i = 1, \ldots, N$ from the observations $\mathbf{Y}_i(k), k = 0, 1, \ldots, K_i + 1$ where K_i stands for the observability index of the pair $(\mathbf{W}, \mathbf{E}_i)$. From the available observations $\mathbf{Y}_i(k), k = 0, 1, \ldots, K_i + 1$, two matrices are generated:

$$\bar{\mathbf{Y}}_i = \begin{pmatrix} \mathbf{Y}_i(0) \\ \mathbf{Y}_i(1) \\ \vdots \\ \mathbf{Y}_i(K_i) \end{pmatrix}; \quad \bar{\bar{\mathbf{Y}}}_i = \begin{pmatrix} \mathbf{Y}_i(1) \\ \mathbf{Y}_i(2) \\ \vdots \\ \mathbf{Y}_i(K_i+1) \end{pmatrix}.$$

Define the block matrix $\begin{bmatrix} \bar{\mathbf{Y}}_i \\ \bar{\bar{\mathbf{Y}}}_i \end{bmatrix} = \tilde{\mathbf{U}}\mathbf{\Sigma}\tilde{\mathbf{V}}^T$ in a singular value decomposition form. From here, the matrix $\tilde{\mathbf{U}}$ of left singular vector of the matrix $(\bar{\mathbf{Y}}_i^T \ \bar{\bar{\mathbf{Y}}}_i^T)^T$ is computed.

Then, we partition $\tilde{\mathbf{U}}$ as $\tilde{\mathbf{U}} = (\tilde{\mathbf{U}}_1^T \ \tilde{\mathbf{U}}_2^T)^T$, the two blocks having the same number of rows.

Computing the matrices $\mathbf{R}_1 = \tilde{\mathbf{U}}_1^T\tilde{\mathbf{U}}_1$, $\mathbf{R}_2 = \tilde{\mathbf{U}}_1^T\tilde{\mathbf{U}}_2$ and $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1^{-1}$, [40] points out that the consensus matrix and the matrix $\mathbf{R}$ built from the observations have exactly the same spectrum. Therefore, $\mathbf{D}$ results from the eigenvalues decomposition of the matrix $(\tilde{\mathbf{U}}_1^T \ \tilde{\mathbf{U}}_2)(\tilde{\mathbf{U}}_1^T \ \tilde{\mathbf{U}}_1)^{-1}$.

Finally, the Laplacian eigenvalues can be obtained by the equation $\mathbf{\Delta} = \frac{1}{\epsilon}(\mathbf{I}_N - \mathbf{D})$.

However, this method is only applicable to networks having nodes with sufficient storage and computation capabilities. Actually, data to be stored equals to $\bar{d}_i M(K_i + 2)$.

In this thesis, the approach is based on the fact that the averaging matrix can be viewed as a product of Laplacian-based consensus matrices with step-sizes given by the inverses of the Laplacian eigenvalues [42]. Therefore, by computing a factorization of the averaging matrix in a distributed way, a distributed estimation of the Laplacian eigenvalues is achieved.

Remark 9

Hereafter, we denote the Laplacian eigenvalue $\lambda_i(\boldsymbol{L})$ as λ_i.

4.3 Distributed solutions to the network robustness

4.3.1 Distributed Laplacian eigenvalues estimation

4.3.1.1 Problem statement

Consider a network modelled by a connected undirected graph $G(V, E)$. Its state $\mathbf{x}(k) = [x_1(k), x_2(k), \ldots, x_N(k)]^T$ evolves as:

$$\mathbf{x}(k+1) = (\mathbf{I}_N - \alpha\mathbf{L})\mathbf{x}(k) = (\mathbf{I}_N - \alpha\mathbf{L})^k\mathbf{x}(0), \tag{4.3.1}$$

where α is selected such as $0 < \alpha < \frac{1}{d_{max}}$ to ensure asymptotic convergence [74]. From the state $\mathbf{x}(k)$, one can define $\mathbf{q}(k) = \mathbf{L}^k\mathbf{x}(0)$ as:

$$\mathbf{q}(k) = (q_{1,k}, \ldots, q_{N,k})^T = \frac{1}{(-\alpha)^k}\left(\mathbf{x}(k) - \sum_{i=0}^{k-1}\binom{k}{i}(-\alpha)^i\mathbf{q}(i)\right), \tag{4.3.2}$$

with $\mathbf{q}(0) = \mathbf{x}(0)$.

1. Assumption 1: The number N of nodes is known.
2. Assumption 2: Each node i stores its initial value $x_i(0)$ and the consecutive values $q_{i,k}, k = 0, \ldots, h$.

The problem under study can therefore be formulated as follows:

Given an arbitrary input-output pair $\{\mathbf{x}(0), \bar{\mathbf{x}} = \bar{x}\mathbf{1}\}$, where $\bar{\mathbf{x}} = \frac{\mathbf{1}\mathbf{1}^T}{N}\mathbf{x}(0)$, and the intermediate observations $\boldsymbol{q}(k), k = 0, \ldots, h$, estimate in a distributed way the Laplacian eigenvalues of the graph representing the network.

However, instead of computing directly the Laplacian eigenvalues, we will compute the inverse of the eigenvalues. In addition, since the number D of distinct nonzero Laplacian eigenvalues is unknown, we will re-parametrize the factorization. Let us first state the following proposition:

Proposition 2

Consider a connected graph with Laplacian matrix $\mathbf{L}$ *and distinct nonzero eigenvalues* $\lambda_2, \cdots, \lambda_{D+1}$. *Assume that the local values vector* $\mathbf{x}(0)$ *is not orthogonal to any eigenvector of* $\mathbf{L}$, *the cost function*

$$E(\boldsymbol{\alpha}) = \|\mathbf{x}(h) - \bar{\mathbf{x}}\|^2 = \left\| \prod_{k=h}^{1} (\boldsymbol{I}_N - \alpha_k \boldsymbol{L})\mathbf{x}(0) - \bar{\mathbf{x}} \right\|^2, \tag{4.3.3}$$

with $\boldsymbol{\alpha} = (\alpha_1 \cdots \alpha_h)^T$, $h \geq D$, *is minimal if and only if* $\{1/\lambda_2, \cdots, 1/\lambda_{D+1}\} \subseteq \{\alpha_1, \alpha_2, \cdots, \alpha_h\}$.

Proof: Consider the eigenvalues decomposition of the Laplacian matrix: $\mathbf{L} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^T$, where $\boldsymbol{\Delta} = diag(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U}$ being the orthogonal matrix of the eigenvectors. Without loss of generality, we assume that the $(D+1)$ first entries of the diagonal matrix $\boldsymbol{\Delta}$ are the $(D+1)$ distinct graph Laplacian eigenvalues. The graph being connected, we know that $\lambda_1 = 0$. Therefore, the corresponding eigenvector (first column of $\mathbf{U}$) is equal to $\frac{1}{\sqrt{N}}\mathbf{1}$. Let $\mathbf{e}_i$ be the i-th vector of the canonical basis of $\mathbb{R}^N$. Then the averaging matrix $\mathbf{J}_N$ is equal to $\mathbf{U}diag(\mathbf{e}_1)\mathbf{U}^T$. We can therefore rewrite the cost function $E(\boldsymbol{\alpha})$ as:

$$\begin{aligned} E(\boldsymbol{\alpha}) &= \left\| \prod_{k=1}^{h} \left(\mathbf{I}_N - \alpha_k \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^T\right)\mathbf{x}(0) - \mathbf{J}_N\mathbf{x}(0) \right\|^2 \\ &= \left\| \mathbf{U}\left(\prod_{k=1}^{h} (\mathbf{I}_N - \alpha_k \boldsymbol{\Delta}) - diag(\mathbf{e}_1) \right) \mathbf{U}^T\mathbf{x}(0) \right\|^2 \\ &= \left\| \mathbf{U}diag\left(\boldsymbol{\epsilon}\right)\mathbf{U}^T\mathbf{x}(0) \right\|^2, \end{aligned}$$

with $\boldsymbol{\epsilon} = (0, \prod_{k=1}^{h}(1-\alpha_k\lambda_2), \cdots, \prod_{k=1}^{h}(1-\alpha_k\lambda_N))^T$. Now, let denote by w_i, $i = 1, \cdots, N$, the entries of $\mathbf{U}^T\mathbf{x}(0)$. Then:

$$E(\boldsymbol{\alpha}) = \sum_{i=2}^{N} \prod_{k=1}^{h} (1-\alpha_k\lambda_i)^2 w_i^2.$$

We can note that this function is a sum of square terms. Therefore, it vanishes if and only if each term $\prod_{k=1}^{h}(1-\alpha_k\lambda_i)^2 w_i^2$, $i = 2, \cdots, N$ is equal to zero. Since, $\mathbf{x}(0)$ is not orthogonal to a Laplacian eigenvector, then $w_i \neq 0$, $\forall i$. Therefore, $E(\boldsymbol{\alpha})$ vanishes if and only if $\prod_{k=1}^{h}(1-\alpha_k\lambda_i)^2 = 0$, $i = 2, \cdots, N$. Due to the multiplicities of the Laplacian eigenvalues, we have in fact D distinct equations $\prod_{k=1}^{h}(1-\alpha_k\lambda_i)^2 = 0$, $i = 2, \cdots, D+1$. These equations are fulfilled if and only if there

are D coefficients α_k respectively equal to the inverse of the D distinct Laplacian eigenvalues. ■

From Proposition 2, we can conclude that the set $\mathbf{S}_1 = \{\alpha_k, k = 1, \ldots, h\}$, of solutions of (4.3.3) contains the inverse of Laplacian eigenvalues $\mathbf{S}_2 = \{\frac{1}{\lambda_i}, i = 2, \ldots, D{+}1\}$: $\mathbf{S}_2 \subset \mathbf{S}_1$. Therefore, the estimation procedure is to be divided in two parts. We first get $\mathbf{S}_1$ by solving (4.3.3) in a distributed way, then we retrieve the set $\mathbf{S}_2$ from $\mathbf{S}_1$ and deduce the Laplacian eigenvalues locally.

We know that if $\{\alpha_k\}_{k=1}^{h}$ contains D inverses of the nonzero Laplacian eigenvalues, then

$$\bar{\mathbf{x}} = \prod_{k=1}^{h}(\mathbf{I} - \alpha_k \mathbf{L})\mathbf{x}(0),$$

or equivalently,

$$\bar{\mathbf{x}} = \sum_{k=0}^{h} c_k \mathbf{L}^k \mathbf{x}(0) = \sum_{k=0}^{h} c_k \mathbf{q}(k) = \mathbf{Q}\mathbf{c}, \tag{4.3.4}$$

where $\mathbf{Q} = [\mathbf{q}(0)\ \mathbf{q}(1)\ \cdots\ \mathbf{q}(h)] \in I\!R^{N\times(h+1)}$ and $\mathbf{c} = [c_0\ c_1\ \cdots\ c_h]^T \in I\!R^{(h+1)\times 1}$, the coefficient c_i being defined as:

$$c_k = \begin{cases} 1, & \text{if } k = 0. \\ (-1)^k \sum_{i<j<\cdots<k} \alpha_i \alpha_j \cdots \alpha_k, & \text{if } k = 1, \ldots, h-1. \\ (-1)^h \prod_{k=1}^{h} \alpha_k, & \text{if } k = h. \end{cases} \tag{4.3.5}$$

It is obvious that α_k are the roots of the polynomial with coefficients c_k, and $\mathbf{c}$ has an alternating sign pattern.

We can state the following proposition which is a corollary of Proposition 2:

Proposition 3

Let $\mathcal{P}(\boldsymbol{c}, :)$ be the h^{th} degree polynomial with coefficients $\boldsymbol{c} = [c_0\ c_1\ \ldots\ c_h]^T \in I\!R^{(h+1)\times 1}$ built from the set of stepsizes $\boldsymbol{S}_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_h\}$ by using (4.3.5). For any node i and intermediate measurements $\boldsymbol{q}_i = [q_{i,0}, q_{i,1}, \ldots, q_{i,h}]^T \in I\!R^{(h+1)\times 1}$ of the consensus protocol (4.3.1), we get

$$\bar{x} = \mathcal{P}(\boldsymbol{c}, \boldsymbol{q}_i) = \boldsymbol{q}_i^T \boldsymbol{c} \tag{4.3.6}$$

if and only if the D inverses of the distinct Laplacian eigenvalues are contained in $\boldsymbol{S}_1$.

Given an upper-bound h for the number D of distinct Laplacian eigenvalues, the idea is to minimize (4.3.3) and then reduce step by step the obtained set of coefficients c_k built from $\mathbf{S}_1$ until it is minimal. After all, the set of the inverses of distinct Laplacian eigenvalues $\mathbf{S}_2$ is achieved. In order to obtain the whole Laplacian spectrum $sp(\mathbf{L})$, the

corresponding multiplicities $m_i, i = 1, \ldots, D+1$ of the distinct Laplacian eigenvalues $\Lambda = \{\lambda_1, \ldots, \lambda_{D+1}\}$ should be estimated (it will be considered in Section 4.3.2).

For a simple understanding of the structure of this Chapter, Figure 4.2 illustrates its organization, showing the way to estimate the network robustness.

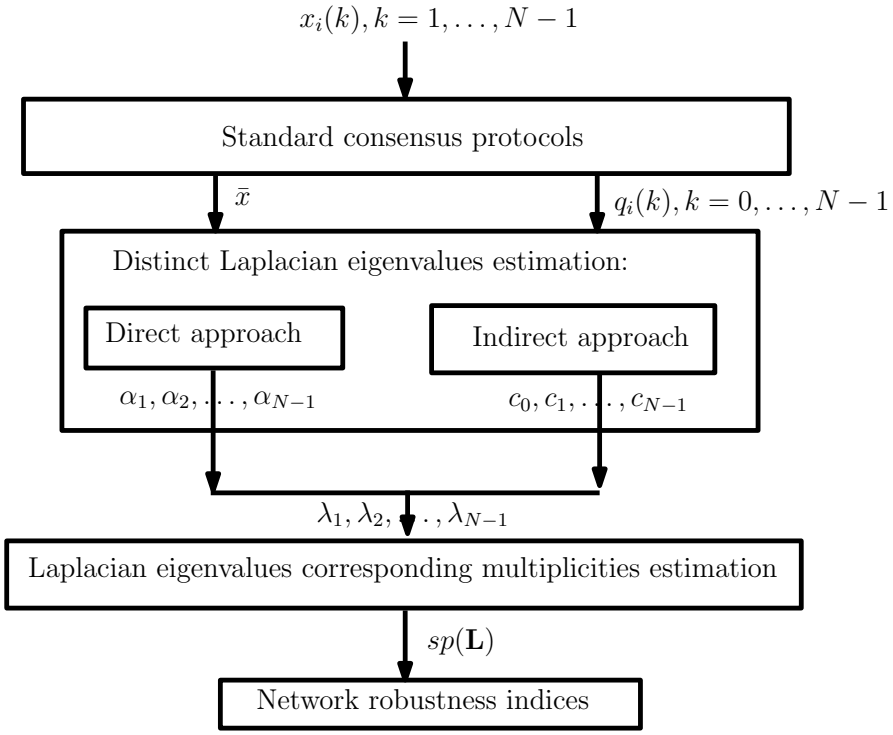


Figure 4.2: Mechanism of network robustness estimation

In fact, for distributed Laplacian eigenvalues estimation, three cases have been considered:

1. The case of perfect knowledge of the average consensus value $\bar{x}$,
2. The case of imperfect knowledge of the average consensus value $\bar{x}$,
3. The case of unknown average consensus value $\bar{x}$.

4.3.1.2 Laplacian-based factorization of the averaging matrix with perfect knowledge of $\bar{x}$

Given an arbitrary input-output pair $\{\mathbf{x}(0), \bar{\mathbf{x}}\}$ and the intermediate observations $\mathbf{q}(k), k = 0, 1, \ldots, h = N-1$, we can minimize the cost function (4.3.3) to get the set of the stepsizes $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{N-1})$, whose inverses are the Laplacian eigenvalues $\{\lambda_1, \ldots, \lambda_{N-1}\}$.

In this way, the proposed direct method with respect to the stepsizes $\boldsymbol{\alpha}$ consists in solving a non-convex optimization (4.3.3) through the Lagrange method of multipliers.

Method 1: Direct distributed factorization of the averaging matrix.

Equation (4.3.3) shows that α_k are global parameters. For distributively carrying out the factorization of the averaging matrix as factors of Laplacian-based consensus matrices, we can first note that the cost function (4.3.3) can be rewritten as:

$$E(\boldsymbol{\alpha}) = \|\mathbf{x}(h) - \bar{\mathbf{x}}\|^2 = \| \prod_{k=h}^{1} (\mathbf{I}_N - diag(\boldsymbol{\alpha}_k)\mathbf{L})\mathbf{x}(0) - \bar{\mathbf{x}}\|^2, \tag{4.3.7}$$

with $\boldsymbol{\alpha}_k = [\alpha_{1,k}, \alpha_{2,k}, \ldots, \alpha_{N,k}]^T = \alpha_k \mathbf{1}, k = 1, 2, \ldots, h = N - 1$.

The idea of the proposed method is to minimize the disagreement between neighbors on the value of α_k while ensuring that the factorization of the averaging matrix is achieved. Such a factorization is assessed by constraining the values of the nodes after h iterations of the consensus algorithm to be equal to the average of the initial values:

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in I\!R^{N\times 1}, k=1,2,\ldots,h} \quad & \frac{1}{2} \sum_{k=1}^{h} \sum_{i \in V} \sum_{j \in N_i} (\alpha_{j,k} - \alpha_{i,k})^2, \\ \text{subject to} \quad & \mathbf{x}(h) = \bar{\mathbf{x}} \end{aligned} \tag{4.3.8}$$

or equivalently,

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in I\!R^{N\times 1}, k=1,\ldots,h} \quad & \frac{1}{2} \sum_{k=1}^{h} \boldsymbol{\alpha}_k^T \mathbf{L} \boldsymbol{\alpha}_k. \\ \text{subject to} \quad & \mathbf{x}(h) = \bar{\mathbf{x}} \end{aligned} \tag{4.3.9}$$

Remark 10

Any solution of (4.3.9) contains the inverse of Laplacian eigenvalues according to Proposition 2.

The constrained optimization problem (4.3.9) can then be performed as an unconstrained optimization problem by means of a Lagrange method with a Lagrange function defined as follows:

$$H(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_h, \mathbf{y}) = \frac{1}{2} \sum_{k=1}^{h} \boldsymbol{\alpha}_k^T \mathbf{L} \boldsymbol{\alpha}_k + \mathbf{y}^T (\mathbf{x}(h) - \bar{\mathbf{x}}) \tag{4.3.10}$$

where $\mathbf{y} \in I\!R^{N\times 1}$ is the vector of Lagrange multipliers.

The minimization of the Lagrange function (4.3.10) can be viewed as h consensus problems to be solved simultaneously with a constraint that provides a kind of external reference to be tracked.

To avoid misunderstanding between time-step k and optimization iteration t, we denote $\mathbf{x}(k), \mathbf{W}(k)$ as $\mathbf{x}_k, \mathbf{W}_k$ respectively.

In order to derive the corresponding distributed algorithm, we now state the following technical lemma:

Lemma 4

The derivatives of the cost function $H(\boldsymbol{\alpha}_k, \mathbf{y})$ defined in (4.3.9) can be computed as follows:

$$\frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k} = \boldsymbol{L}\boldsymbol{\alpha}_k - diag^{-1}(\boldsymbol{\alpha}_k) diag(\mathbf{x}_{k-1} - \mathbf{x}_k)\boldsymbol{\delta}_k, \tag{4.3.11}$$

where

$$\boldsymbol{\delta}_h = \mathbf{y} \text{ and } \boldsymbol{\delta}_{k-1} = \boldsymbol{W}_k\boldsymbol{\delta}_k, \quad k = 1, \ldots, h. \tag{4.3.12}$$

Proof: The consensus network being a linear system we know that $\mathbf{x}_k = \mathbf{W}_k\mathbf{x}_{k-1}$. Therefore, we can explicitly write the output according to the weighting matrix of interest, i.e. $\mathbf{x}_h = \mathbf{W}_h\mathbf{x}_{h-1}$ and $\mathbf{x}_h = \prod_{j=h}^{k+1}\mathbf{W}_j\mathbf{W}_k\mathbf{x}_{k-1}$, $k = 1, \cdots, h-1$.

The cost function can be written as:

$$\begin{aligned} H(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_h, \mathbf{y}) &= \frac{1}{2}\sum_{k=1}^{D}\boldsymbol{\alpha}_k^T\mathbf{L}\boldsymbol{\alpha}_k + \mathbf{y}^T(\prod_{i=h}^{k+1}\mathbf{W}_i\mathbf{W}_k\mathbf{x}_{k-1} - \bar{\mathbf{x}}) \\ &= \frac{1}{2}\sum_{k=1}^{h}\boldsymbol{\alpha}_k^T\mathbf{L}\boldsymbol{\alpha}_k + \mathbf{y}^T(\prod_{i=h}^{k+1}\mathbf{W}_i(\mathbf{I}_N - \mathrm{diag}(\boldsymbol{\alpha}_k)\mathbf{L}\mathbf{x}_{k-1} - \bar{\mathbf{x}}) \\ &= \frac{1}{2}\sum_{k=1}^{h}\boldsymbol{\alpha}_k^T\mathbf{L}\boldsymbol{\alpha}_k + \mathbf{y}^T(\prod_{i=h}^{k+1}\mathbf{W}_i\mathbf{x}_{k-1} - \bar{\mathbf{x}}) - \mathbf{y}^T\prod_{i=h}^{k+1}\mathbf{W}_i\mathrm{diag}(\boldsymbol{\alpha}_k)\mathbf{L}\mathbf{x}_{k-1}). \end{aligned}$$

Now, using the property (A.0.1) of the Khatri-Rao product, we get:

$$H(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_h, \mathbf{y}) = \frac{1}{2}\sum_{k=1}^{h}\boldsymbol{\alpha}_k^T\mathbf{L}\boldsymbol{\alpha}_k + \mathbf{y}^T(\prod_{i=h}^{k+1}\mathbf{W}_i\mathbf{x}_{k-1} - (\mathbf{x}_{k-1}^T\mathbf{L}^T \odot \prod_{i=h}^{k+1}\mathbf{W}_i)\boldsymbol{\alpha}_k - \bar{\mathbf{x}}).$$

Therefore, we can easily deduce that

$$\frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k} = \mathbf{L}\boldsymbol{\alpha}_k - (\mathbf{x}_{k-1}^T\mathbf{L}^T \odot \prod_{i=h}^{k+1}\mathbf{W}_i)^T\mathbf{y}.$$

Taking the symmetry of the consensus matrices into account, we get:

$$\begin{aligned} \frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k} &= \mathbf{L}\boldsymbol{\alpha}_k - \mathrm{diag}(\mathbf{x}_{k-1}^T\mathbf{L}^T)^T\prod_{i=k+1}^{h}\mathbf{W}_i\mathbf{y} \\ &= \mathbf{L}\boldsymbol{\alpha}_k - \mathrm{diag}(\mathbf{L}\mathbf{x}_{k-1})\prod_{i=k+1}^{h-2}\mathbf{W}_i\mathbf{W}_{h-1}\underbrace{\mathbf{W}_h\mathbf{y}}_{\boldsymbol{\delta}_{h-1}} \end{aligned}$$

$$= \mathbf{L}\boldsymbol{\alpha}_k - \text{diag}(\mathbf{L}\mathbf{x}_{k-1}) \prod_{i=k+1}^{D-2} \mathbf{W}_i \underbrace{\mathbf{W}_{h-1}\boldsymbol{\delta}_{h-1}}_{\boldsymbol{\delta}_{h-2}}$$

$$= \mathbf{L}\boldsymbol{\alpha}_k - \text{diag}(\mathbf{L}\mathbf{x}_{k-1})\boldsymbol{\delta}_k.$$

Since $\mathbf{x}_k = (\mathbf{I}_N - diag(\boldsymbol{\alpha}_k)\mathbf{L})\mathbf{x}_{k-1}$, we can deduce that $\mathbf{L}\mathbf{x}_{k-1} = \text{diag}^{-1}(\boldsymbol{\alpha}_k)(\mathbf{x}_{k-1} - \mathbf{x}_k)$. Hence, $\text{diag}(\mathbf{L}\mathbf{x}_{k-1}) = \text{diag}^{-1}(\boldsymbol{\alpha}_k)\text{diag}(\mathbf{x}_{k-1} - \mathbf{x}_k)$. As a consequence,

$$\frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k} = \mathbf{L}\boldsymbol{\alpha}_k - \text{diag}^{-1}(\boldsymbol{\alpha}_k)\text{diag}(\mathbf{x}_{k-1} - \mathbf{x}_k)\boldsymbol{\delta}_k.$$

■

Applying the results of Lemma 4, the updating scheme of the optimization algorithm is as follows:

$$\begin{aligned}\boldsymbol{\alpha}_k[t+1] &= \boldsymbol{\alpha}_k[t] - \beta\frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k[t]} \\ &= (\mathbf{I}_N - \beta\mathbf{L})\boldsymbol{\alpha}_k[t] + \beta\text{diag}^{-1}(\boldsymbol{\alpha}_k)\text{diag}(\mathbf{x}_{k-1}[t] - \mathbf{x}_k[t])\boldsymbol{\delta}_k) \qquad (4.3.13)\\ \mathbf{y}[t+1] &= \mathbf{y}[t] + \mu(\mathbf{x}_h[t] - \bar{\mathbf{x}}). \qquad (4.3.14)\end{aligned}$$

The proposed distributed algorithm can then be described as follows:

Algorithm 4 (***Lagrange multipliers method for distributed averaging matrix factorization***)

Inputs:

- *An upper-bound h of the number of distinct Laplacian eigenvalues $h = N - 1$,*
- *learning rates $0 < \mu < 1$ and $0 < \beta < 1$,*
- *initial input-output values $\{x_i(0), \bar{x}_i\}, i = 1, \ldots, N$ with $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i(0)$ obtained from standard consensus protocols.*

1. *Initialization: Each node $i, i = 1, \ldots, N$ generates random steps-sizes $\alpha_{i,k}[0], k = 1, \ldots, h$ and Lagrange multipliers $y_i[0]$ and then set $t = 0$.*
2. *Propagation of Lagrange multipliers:*
   (a) *Set: $\delta_{i,h}[t] = y_i[t]$*
   (b) *Propagate the Lagrange multipliers for $k = h, \cdots, 2$:*
   $$\delta_{i,k-1}[t] = \delta_{i,k}[t] + \sum_{j \in N_i}(\delta_{j,k}[t] - \delta_{i,k}[t]).$$
3. *Update of the factor matrices step-sizes:*
   (a) *$x_{i,0}[t] = x_i(0)$.*

(b) *For $k = 1, 2, \cdots, h$, each node sends a message $m_{i,k}[t]$ containing its current local value $x_{i,k}[t]$, and the local step-size $\alpha_k[t]$.*

(c) *After receiving the messages $m_{j,k}[t]$ from its neighbors $j \in N_i$, each node i carries out the following updates:*

$$x_{i,k} = x_{i,k-1} + \alpha_{i,k} \sum_{j \in N_i} (x_{j,k-1} - x_{i,k-1})$$

$$\alpha_{i,k}[t+1] = \alpha_{i,k}[t] - \beta(\sum_{j \in N_i} (\alpha_{j,k}[t] - \alpha_{k,i}[t])) - \beta \sum_{j \in N_i} (x_{j,k-1}[t] - x_{i,k-1}[t])\delta_{i,k}[t]$$

4. *Update the Lagrange multiplier using (4.3.13)*

5. *Exit if a stopping criterion is reached else set $t := t + 1$ and return to 2.*

Outputs: Set $\mathbf{S}_{1,i}$ of step-size $\alpha_{i,k}[t]$.
Each node deduces the Laplacian eigenvalues as $\lambda_{i,k} = \frac{1}{\alpha_{i,k}[t+1]}$.

This algorithm is similar to a gradient back-propagation algorithm. According to [14], selecting β, μ in the interval of $(0, 1)$ ensures convergence to a local minimum as long as the constraint $\mathbf{x}(h) = \bar{\mathbf{x}}$ and the constraint on α_k are achieved. The obtained solution contains the inverse of the Laplacian eigenvalues. The speed of convergence depends on the choice of these parameters.

In order to accelerate the speed of convergence, we can re-parameterize the problem (4.3.1) by means of the equation (4.3.4).

$$\bar{\mathbf{x}} = \sum_{k=0}^{h} c_k \mathbf{L}^k \mathbf{x}(0) = \sum_{k=0}^{h} c_k \mathbf{q}(k) = \mathbf{Q}\mathbf{c}.$$

Instead of the minimization with respect to the step-sizes α_k, the new optimization with respect to the polynomial coefficients $c_k, k = 0, \ldots, h$ can be applied. That is called indirect distributed factorization of the averaging matrix.

Method 2: Indirect distributed factorization of the averaging matrix.

The aim of this method is to find the polynomial coefficients c_k in a distributed way by solving the problem:

$$\min_{\mathbf{c}} \|\bar{\mathbf{x}} - \mathbf{Q}\mathbf{c}\|^2,$$

with $\mathbf{Q} = [\mathbf{q}(0)\ \mathbf{q}(1)\ \cdots\ \mathbf{q}(h)] \in I\!R^{N \times (h+1)}$.

All the initial conditions $\mathbf{x}(0)$ should not be eigenvectors of Laplacian matrix $\mathbf{L}$ and not be identical, meaning that $\mathbf{x}(0) \neq \delta\mathbf{1}, \delta \in I\!R$, otherwise all the columns of $\mathbf{Q}$ are zero

except the first one. Since the initial condition is randomly generated, almost surely it cannot be an eigenvector of Laplacian matrix $\mathbf{L}$.

Lemma 5

Consider a connected undirected graph $G(V, E)$ with Laplacian spectrum $sp(\boldsymbol{L}) = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$. Herein, $sp(\boldsymbol{L})$ has $D+1$ distinct Laplacian eigenvalues. If $\boldsymbol{x}(0)$ is not an eigenvector of $\boldsymbol{L}$, then the matrix

$$\boldsymbol{Q} = \boldsymbol{U} diag(\boldsymbol{U}^T \boldsymbol{x}(0)) \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^h \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^h \\ & & \cdots\cdots & & \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^h \end{pmatrix}$$

has rank $min(D+1, N)$, where $\boldsymbol{U}$ is the matrix of Laplacian eigenvectors.

Proof:

$$\mathbf{Q} = [\mathbf{q}(0)\ \mathbf{q}(1)\ \ldots \mathbf{q}(h)] \in I\!R^{N \times (h+1)}, \quad \text{where } \mathbf{q}(k) = \mathbf{L}^k \mathbf{x}(0).$$

Equivalently,

$$\begin{aligned} \mathbf{Q} &= [\mathbf{I}_N \mathbf{x}(0)\ \mathbf{L}\mathbf{x}(0)\ \mathbf{L}^2\mathbf{x}(0) \ldots \mathbf{L}^h\mathbf{x}(0)] \\ &= [\mathbf{I}_N\ \mathbf{L}\ \mathbf{L}^2 \ldots\ \mathbf{L}^h] diag(\underbrace{\mathbf{x}(0), \ldots, \mathbf{x}(0)}_{h+1 \text{ times}}] \end{aligned}$$

Knowing that $\mathbf{L} = \mathbf{U}\mathbf{\Delta}\mathbf{U}^T$, where $\mathbf{\Delta} = diag(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U}$ are the diagonal matrix of Laplacian eigenvalues and the orthogonal matrix of the eigenvectors respectively. Hence,

$$[\mathbf{I}_N\ \mathbf{L}\ \mathbf{L}^2 \ldots\ \mathbf{L}^h] = \mathbf{U}[\mathbf{I}_N\ \mathbf{\Delta}\ \mathbf{\Delta}^2 \ldots\ \mathbf{\Delta}^h] diag(\underbrace{\mathbf{U}^T, \ldots, \mathbf{U}^T}_{h+1 \text{ times}})$$

Therefore,

$$\mathbf{Q} = \mathbf{U}[\mathbf{I}_N\ \mathbf{\Delta}\ \mathbf{\Delta}^2 \ldots\ \mathbf{\Delta}^h] diag(\underbrace{\mathbf{U}^T\mathbf{x}(0), \ldots, \mathbf{U}^T\mathbf{x}(0)}_{h+1 \text{ times}})$$

where $diag(\mathbf{U}^T\mathbf{x}(0), \ldots, \mathbf{U}^T\mathbf{x}(0)) \in I\!R^{((h+1)\times N)\times(h+1)}$.

Denote by $\hat{\mathbf{\Lambda}} = [\lambda_1, \ldots, \lambda_N]^T$ the vector of Laplacian eigenvalues. Therefore,

$$\begin{aligned} \mathbf{Q} &= \mathbf{U} \begin{pmatrix} \mathbf{1}_N \circ \mathbf{U}^T\mathbf{x}(0) \\ \hat{\mathbf{\Lambda}} \circ \mathbf{U}^T\mathbf{x}(0) \\ \hat{\mathbf{\Lambda}}^2 \circ \mathbf{U}^T\mathbf{x}(0) \\ \cdots\cdots \\ \hat{\mathbf{\Lambda}}^h \circ \mathbf{U}^T\mathbf{x}(0) \end{pmatrix}^T = \mathbf{U} \begin{pmatrix} diag(\mathbf{U}^T\mathbf{x}(0))\mathbf{1}_N) \\ diag(\mathbf{U}^T\mathbf{x}(0))\hat{\mathbf{\Lambda}} \\ diag(\mathbf{U}^T\mathbf{x}(0))\hat{\mathbf{\Lambda}}^2 \\ \cdots\cdots\cdots \\ diag(\mathbf{U}^T\mathbf{x}(0))\hat{\mathbf{\Lambda}}^h \end{pmatrix}^T \\ &= \mathbf{U} diag(\mathbf{U}^T\mathbf{x}(0)) \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^h \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^h \\ & & \cdots\cdots & & \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^h \end{pmatrix} = \mathbf{UFB} \end{aligned}$$

where $\circ$ denotes the Hamadard product.

Besides that, $h = N - 1$, hence the Vandermonde matrix $\mathbf{B}$ is a square matrix. Therefore, $Rank(\mathbf{Q}) = min(rank(\mathbf{F}), rank(\mathbf{B}))$.

Here, if $\mathbf{U}^T\mathbf{x}(0) \neq \mathbf{0}$, or equivalently $\mathbf{x}(0)$ is not an eigenvector of the Laplacian matrix, then $rank(\mathbf{F}) = rank(diag(\mathbf{U}^T\mathbf{x}(0))) = N$. On the other hand, since $\mathbf{B}$ is a Vandermonde matrix, one property can be reviewed is that rank of the Vandermonde is equal to the number of distinct generators (Laplacian eigenvalues), meaning that $rank(\mathbf{B}) = D + 1$. Thus, $rank(\mathbf{Q}) = D + 1$.

From the other side, if there exist elements of the vector $\mathbf{U}^T\mathbf{x}(0)$ equal to 0, then $rank(\mathbf{Q}) = min(D + 1, l)$ with l being the number of nonzero elements of $\mathbf{U}^T\mathbf{x}(0)$. ■

Remark 11

$rank(\boldsymbol{Q}) < N$ if there exists $\lambda_i \in sp(\boldsymbol{L})$ with multiplicity higher than 1, $\forall \mathbf{x}(0)$.

In a centralized way, in order to find $\mathbf{c}$ from the equation $\mathbf{Qc} = \bar{\mathbf{x}}$, $\mathbf{Q}$ must be a full rank matrix. Then,

$$\mathbf{c} = \mathbf{Q}^{-1}\bar{\mathbf{x}}$$

In constrast, if $\mathbf{Q}$ is not a full rank matrix, then there are several solutions for $\mathbf{c}$. To deal with this situation, we make use of a least-square solution to find the most common solution S_1 inside the set of solutions $(So_1, So_2, \ldots, So_m)$ of $\mathbf{c}$ as described in Figure 4.3.

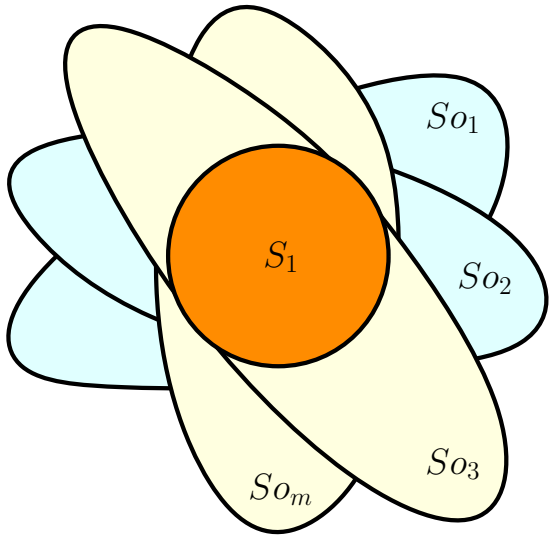


Figure 4.3: Several solutions of $\mathbf{c}$ in the case of a not-a-full-rank matrix $\mathbf{Q}$.

The problem now becomes:

$$\min_{\mathbf{c}} \|\mathbf{c}\|^2 \text{ subject to } \mathbf{Qc} = \bar{\mathbf{x}}$$

where $\mathbf{Q}$ is a full-row-rank matrix.

In order to solve this problem in a distributed way, we solve the problem locally according to $\mathbf{c}_i, i = 1, \ldots, N$. Denoting by $\mathbf{c}_i$ the local version of $\mathbf{c}$ and $\mathbf{Q}_i \in I\!R^{(d_i+1)\times(h+1)}$ the submatrix of $\mathbf{Q}$ constituted with the rows associated with node i and its d_i neighbors, the solution of the previous problem can be obtained in a distributed way by solving

$$\min_{\mathbf{c}_i} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 \quad s.t. \quad \mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}, \ i = 1, \cdots, N, \tag{4.3.15}$$

provided that $\mathbf{Q}_i$ is full row rank.

Let us call $\mathbf{Se}_i$ a row selection matrix with respect to node i. $\mathbf{Se}$ is a matrix, where $Se_{j,j} = 1$ if $j \in (i \cup N_i)$. $\mathbf{Q}_i$ can be expressed as:

$$\mathbf{Q}_i = \mathbf{Se}_i \mathbf{Q} = \mathbf{Se}_i \mathbf{U} diag(\mathbf{U}^T \mathbf{x}(0)) \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^h \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^h \\ & \cdots\cdots & & & \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^h \end{pmatrix}$$

Obviously, $rank(\mathbf{Se}_i \mathbf{U}) = d_i + 1$. As shown in (4.3.15), $\mathbf{Q}_i$ should be a full row rank matrix to ensure the constraint $\mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}$ to be satisfied. Therefore, we can construct $\mathbf{Q}_i = \begin{bmatrix} \mathbf{q}_i \\ \mathbf{q}_j \end{bmatrix}$, $j \in N_i$ by selecting the $\mathbf{q}_j, j \in N_i$ until the $rank(\mathbf{Q}_i)$ cannot increase more.

There is a lot of rules that can be applied for the choice of $\mathbf{q}_j$. However, in our study, we select $\mathbf{q}_j$ based on the condition number of the established $\mathbf{Q}_i$. Meaning that, $\mathbf{q}_j$ is selected so that the condition number of $\mathbf{Q}_i$ is the smallest.

From (4.3.5), knowing that the coefficients c_k have an alternating sign pattern, the problem can be reformulated as follows:

$$\min_{\mathbf{c}_i \in I\!R^{(h+1)}, i=1,\ldots,N} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 . \tag{4.3.16}$$

$$\text{subject to} \quad \mathbf{c}_i \in C_i = \{\mathbf{c}_i \in I\!R^{(h+1)} : \mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}\}, \ i = 1, 2, \cdots, N \tag{4.3.17}$$

$$c_{i,k} = \text{sign}(k) c_{i,k}, \tag{4.3.18}$$

where $\text{sign}(k) = \begin{cases} -1 & \text{if } k \text{ is odd,} \\ 1 & \text{if } k \text{ is even.} \end{cases}$

The problem here is to optimize the sum of convex objective functions corresponding to the nodes in the network. In contrast to the previous direct method, the optimization

problem (4.3.16) is convex. To solve it we will resort to a projected subgradient method inspired from [57] and an alternating direction method of multipliers [11, 22].

(A) *Distributed projected subgradient method*

The main idea of the distributed projected subgradient method is to allow each node i to update its estimate following two steps.

Firstly, a subgradient step is taken by minimizing the local objective function $\frac{1}{2}\sum_{j\in N_i}\|\mathbf{c}_i-\mathbf{c}_j\|^2$. Then, the intermediate result is projected on the constraint set C_i. Therefore, the first step yields:

$$\hat{\mathbf{c}}_i[t+1]=\mathbf{c}_i[t]-\gamma[t]\sum_{j\in N_i}\left(\mathbf{c}_i[t]-\mathbf{c}_j[t]\right), \tag{4.3.19}$$

where $\gamma[t]>0$ is a stepsize, while the second step requires the computation of the projection $\mathbf{c}_i[t+1]=\Omega_{C_i}[\hat{\mathbf{c}}_i[t+1]]$. This projection step is equivalent to solve a constrained optimization problem formulated as:

$$\min \quad \frac{1}{2}\|\mathbf{c}_i[t+1]-\hat{\mathbf{c}}_i[t+1]\|^2 \text{ s.t.} \quad \mathbf{Q}_i\mathbf{c}_i[t+1]=\bar{x}\mathbf{1}.$$

To solve this constrained optimization problem, generally, the Lagrangian augmented function is defined as follows:

$$L(\mathbf{c}_i[t+1],\mathbf{y})=\frac{1}{2}\|\mathbf{c}_i[t+1]-\hat{\mathbf{c}}_i[t+1]\|^2+\mathbf{y}^T(\mathbf{Q}_i\mathbf{c}_i[t+1]-\bar{x}\mathbf{1}).$$

Following the KKT (Karush-Kuhn-Tucker) conditions, we have to solve:

$$\nabla_{\mathbf{c}_i[t+1]}L(\mathbf{c}_i[t+1],\mathbf{y})=\mathbf{0}. \tag{4.3.20}$$

$$\nabla_{\mathbf{y}}L(\mathbf{c}_i[t+1],\mathbf{y})=\mathbf{0}. \tag{4.3.21}$$

Let solve (4.3.20) first:

$$\begin{aligned} &\mathbf{c}_i[t+1]-\hat{\mathbf{c}}_i[t+1]+\mathbf{Q}_i^T\mathbf{y}=\mathbf{0} \\ \Leftrightarrow \quad &\mathbf{c}_i[t+1]=\hat{\mathbf{c}}_i[t+1]-\mathbf{Q}_i^T\mathbf{y}. \end{aligned} \tag{4.3.22}$$

Then, we substitue $\mathbf{c}_i$ into (4.3.21):

$$\begin{aligned} &\mathbf{Q}_i\mathbf{c}_i[t+1]-\bar{x}\mathbf{1}=\mathbf{0} \\ \Leftrightarrow \quad &\mathbf{Q}_i(\hat{\mathbf{c}}_i[t+1]-\mathbf{Q}_i^T\mathbf{y})=\bar{x}\mathbf{1} \\ \Leftrightarrow \quad &\mathbf{Q}_i\hat{\mathbf{c}}_i[t+1]-\mathbf{Q}_i\mathbf{Q}_i^T\mathbf{y}=\bar{x}\mathbf{1} \\ \Leftrightarrow \quad &\mathbf{y}=(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}(\mathbf{Q}_i\hat{\mathbf{c}}_i[t+1]-\bar{x}\mathbf{1}). \end{aligned} \tag{4.3.23}$$

Now, we substitute (4.3.23) into (4.3.22), we get:

$$\mathbf{c}_i[t+1]=\hat{\mathbf{c}}_i[t+1]-\mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}(\mathbf{Q}_i\hat{\mathbf{c}}_i[t+1]-\bar{x}\mathbf{1})$$

$$\Leftrightarrow \quad \mathbf{c}_i[t+1] = \hat{\mathbf{c}}_i[t+1] - \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}\mathbf{Q}_i\hat{\mathbf{c}}_i[t+1] + \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}\bar{x}\mathbf{1}$$
$$\Leftrightarrow \quad \mathbf{c}_i[t+1] = (\mathbf{I}_{h+1} - \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}\mathbf{Q}_i)\hat{\mathbf{c}}_i[t+1] + \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}\bar{x}\mathbf{1}. \tag{4.3.24}$$

or equivalently,

$$\mathbf{c}_i[t+1] = \tilde{\mathbf{Q}}_i\bar{\mathbf{x}}_i + (\mathbf{I}_{h+1} - \tilde{\mathbf{Q}}_i\mathbf{Q}_i)\hat{\mathbf{c}}_i[t+1], \tag{4.3.25}$$

with $\tilde{\mathbf{Q}}_i = \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$, provided $h \geq d_i$ and $\mathbf{Q}_i$ full row rank. Then, we project $\mathbf{c}_i[t+1]$ on the sign constraint as:

$$c_{i,k}[t+1] = \begin{cases} \max(0, c_{i,k}[t+1]) & \text{if } mod(k,2) = 0. \\ \min(0, c_{i,k}[t+1]) & \text{if } mod(k,2) \neq 0. \end{cases} \tag{4.3.26}$$

The corresponding distributed projected subgradient algorithm is then described in Algorithm 5:

Algorithm 5 (*Projected subgradient algorithm*)

Given $h = N-1$, where N stands for the number of nodes, and a learning rate $0 < \gamma_0 < 1$, the following steps are to be carried out:

1. *Initialization:*
   - *Initial input-output values $\{x_i(0), \bar{x}_i\}$, $i = 1, \cdots, N$, with $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i(0)$ obtained from a standard average consensus algorithm.*
   - *Each node i forms the vector $\mathbf{q}_i$, and the matrices $\mathbf{Q}_i$ and $\tilde{\mathbf{Q}}_i$.*
   - *Each node i generates random initial values for $\mathbf{c}_i[0]$.*
2. *Set $t = 0$;*
   (a) *Set $t := t+1$;*
   (b) *Update the learning rate $\gamma[t]$ by using the diminishing rule: $\gamma[t] = \frac{\gamma_0}{\sqrt{t}}$*
   (c) *Each node updates its vector of parameters $\mathbf{c}_i[t]$ using (4.3.19), (4.3.25) and (4.3.26).*
   (d) *Return to 2a or stops the iterations if a stopping criterion is reached.*
3. *Each node builds a polynomial p_i with the entries of $\mathbf{c}_i[t]$ and compute the set of stepsizes $\{\alpha_{i,k}\}$ as the roots of the polynomial p_i.*

In [57], the convergence of this method has been well studied. With an appropriate choice of the stepsize $\gamma[0]$, Algorithm 5 will converge to a global minimum. The speed of convergence depends on the choice of the gradient stepsize.

(B) *Alternating direction method of multipliers (ADMM).*

ADMM is relatively easy to implement and its convergence properties have been well studied in recent works such that in [8, 11, 23]. The pros of this method is that it is guaranteed to converge for any choice of the tunning parameters under mild conditions [11] while other techniques such as subgradient projection or dual decomposition depend on the choice of the stepsize for the gradient updates. Unlike other methods, ADMM has only one single penalty parameter ρ that can influence on the speed of convergence. ADMM assures a very good convergence speed when its parameters are appropriately chosen. There exist some works dealing with penalty parameter selection for accelerating the convergence rate of ADMM [8, 19, 31, 66, 71]. The convergence analysis is also studied in [8, 11, 19, 23, 25, 66]. However, in this study, our penalty parameter is a fixed constant.

The ADMM algorithm acts by introducing some auxiliary variables $\mathbf{z}_{ij}$, so that a strictly equivalent optimization problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{c}_i \in I\!R^{(h+1)}, i=1,\ldots,N} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i - \mathbf{c}_j\|^2 . \\ \text{subject to} \quad & \mathbf{c}_i = \mathbf{z}_{ij},\ j \in N_i, \\ & \mathbf{z}_{ji} = \mathbf{z}_{ij}, \\ & z_{ij}^k = sign(k) z_{ij}^k, \\ & \mathbf{c}_i \in C_i = \{\mathbf{c}_i \in I\!R^{h+1} : \mathbf{Q}_i\mathbf{c}_i = \bar{x}\mathbf{1}\},\ \ i = 1, 2, \cdots, N. \end{aligned} \tag{4.3.27}$$

where $sign(k) = \begin{cases} 1 & \text{if } k \text{ is even}, \\ -1 & \text{if } k \text{ is odd} \end{cases}$.

The corresponding augmented Lagrangian function is given by:

$$L_\rho(\mathbf{c}, \mathbf{z}, \mathbf{y}) = \sum_{i=1}^{N} L_\rho^{(i)}(\mathbf{c}_i, \mathbf{z}, \mathbf{y}),$$

with

$$L_\rho^{(i)}(\mathbf{c}_i, \mathbf{z}, \mathbf{y}) = \sum_{j\in N_i}\left(\frac{1}{2}\|\mathbf{c}_i - \mathbf{c}_j\|^2 + \mathbf{y}_{ij}^T(\mathbf{c}_i - \mathbf{z}_{ij}) + \frac{\rho}{2}\|\mathbf{c}_i - \mathbf{z}_{ij}\|^2\right), \tag{4.3.28}$$

where ρ is a penalty coefficient. In contrast to the method of Lagrange multipliers, in ADMM the minimization of the cost function according to the primal variables $\mathbf{c}_i$ and $\mathbf{z}_{ij}$ is carried out sequentially. So, given the primal variables and the dual variables $\mathbf{y}_{ij}$, the cost function is first minimized according to $\mathbf{c}_i$ and the obtained solution projected on the constrained set (c-minimization). Then $\mathbf{z}_{ij}$ are obtained from the given dual variables and the updated values of $\mathbf{c}_i$ (z-minimization). The third step corresponds to the update of the dual variables from the updated primal ones (y-minimization):

$$\mathbf{c}_i[t+1] \quad = \Omega_{C_i}[\text{argmin} \quad L_\rho(\hat{\mathbf{c}}_i[t+1], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t])]; \tag{4.3.29}$$

$$\mathbf{z}_{ij}[t+1] = \text{argmin} \quad L_\rho(\mathbf{c}_i[t+1], \mathbf{z}_{ij}, \mathbf{y}_{ij}[t])]; \tag{4.3.30}$$
$$\mathbf{y}_{ij}[t+1] = \text{argmin} \quad L_\rho(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_{ij}[t])]. \tag{4.3.31}$$

These iterations indicate that the method is particularly useful when the $c-$ and $z-$ minimizations can be carried out efficiently. First, we solve the c-minimization:

$$\nabla_{\mathbf{c}_i} L_\rho(\hat{\mathbf{c}}_i[t+1], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t]) = \mathbf{0}.$$

$$\Leftrightarrow \sum_{j\in N_i}(\hat{\mathbf{c}}_i[t+1] - \mathbf{c}_j[t]) + \sum_{j\in N_i}\mathbf{y}_{ij}[t] + \rho(\hat{\mathbf{c}}_i[t+1] - \mathbf{z}_{ij}[t]) = \mathbf{0}$$
$$\Leftrightarrow (d_i(1+\rho))\hat{\mathbf{c}}_i[t+1] = \sum_{j\in N_i}\mathbf{c}_j[t] + \rho\sum_{j\in N_i}\mathbf{z}_{ij}[t] - \sum_{j\in N_i}\mathbf{y}_{ij}[t]$$
$$\Leftrightarrow \hat{\mathbf{c}}_i[t+1] = (d_i(1+\rho))^{-1}(\sum_{j\in N_i}\mathbf{c}_j[t] + \rho\sum_{j\in N_i}\mathbf{z}_{ij}[t] - \sum_{j\in N_i}\mathbf{y}_{ij}[t]),$$

where d_i is the degree of node i.

For the cost function considered herein, we can note that the c-minimization problem (4.3.29) is solved by:

$$\hat{\mathbf{c}}_i[t+1] = (d_i(1+\rho))^{-1}(\sum_{j\in N_i}\mathbf{c}_j[t] + \rho\sum_{j\in N_i}\mathbf{z}_{ij}[t] - \sum_{j\in N_i}\mathbf{y}_{ij}[t]), \tag{4.3.32}$$

d_i being the degree of node i. Then, as for the projected subgradient case we get the same expression for $\mathbf{c}_i[t+1]$ as in (4.3.25).

Next, we solve the sub-optimization (4.3.30) analogically, meaning that $\nabla_{\mathbf{z}_{ij}} L(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_i[t]) = 0$ with respect to the term that $\mathbf{z}_{ij} = \mathbf{z}_{ji}$. Rewrite the augmented Lagrange function:

$$L_\rho(\mathbf{c}_i, \mathbf{z}, \mathbf{y}) = \sum_{j\in N_i}\left(\frac{1}{2}\|\mathbf{c}_i - \mathbf{c}_j\|^2 + \mathbf{y}_{ij}^T(\mathbf{c}_i - \mathbf{z}_{ij}) + \frac{\rho}{2}\|\mathbf{c}_i - \mathbf{z}_{ij}\|^2 + \mathbf{y}_{ji}^T(\mathbf{c}_j - \mathbf{z}_{ji}) + \frac{\rho}{2}\|\mathbf{c}_j - \mathbf{z}_{ji}\|^2\right),$$

Since $\nabla_{\mathbf{z}_{ij}} L(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_i[t]) = \mathbf{0}$ with $\mathbf{z}_{ij} = \mathbf{z}_{ji}$, it is equivalent to:

$$-(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]) - \rho(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + 2\rho\mathbf{z}_{ij}[t+1] = \mathbf{0}$$
$$\Leftrightarrow \quad \mathbf{z}_{ij}[t+1] = \tfrac{1}{2}(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + \tfrac{1}{2\rho}(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]). \tag{4.3.33}$$

Then, we project $\mathbf{z}_{ij}[t+1]$ on the sign constraint as:

$$z_{ij}^k[t+1] = \begin{cases} \max(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) = 0. \\ \min(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) \neq 0. \end{cases} \tag{4.3.34}$$

The Lagrange multipliers are updated as:

$$\mathbf{y}_{ij}[t+1] = \mathbf{y}_{ij}[t] + \rho(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]). \tag{4.3.35}$$

This fully decentralized algorithm is summarized as follows:

Algorithm 6 (*Distributed ADMM algorithm*)

Given $h = N - 1$, where N stands for the number of nodes N, and a penalty parameter ρ, the following steps are to be carried out:

1. *Initialization:*
   - *Initial input-output values $\{x_i(0), \bar{x}_i\}$, $i = 1, \cdots, N$, with $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i(0)$ obtained from a standard average consensus algorithm,*
   - *Each node i forms the vector $\mathbf{q}_i$, and the matrices $\mathbf{Q}_i$ and $\tilde{\mathbf{Q}}_i$.*
   - *Each node i generates random values for $\mathbf{c}_i[0]$, $\mathbf{z}_{ij}[0]$ and $\mathbf{y}_{ij}[0]$, $j \in N_i$.*
2. *Set $t = 0$;*
   (a) *Set $t := t + 1$;*
   (b) *Update equations: each node i*
   - *updates its primary variable $\mathbf{c}_i[t]$ using (4.3.32) and (4.3.25).*
   - *updates the primary variables $\mathbf{z}_{ij}[t]$ associated with its neighbors using (4.3.33) and (4.3.34).*
   - *update the dual variables $\mathbf{y}_{ij}$ using (4.3.35).*
   - *Return to 2a until a stopping criterion is reached.*

   (c) *Each node builds a polynomial p_i with the entries of $\mathbf{c}_i[t]$ and compute the set of the stepsizes $\{\alpha_{i,k}\}$ as the roots of the polynomial p_i.*

The convergence analysis for distributed quadratic programming has been studied deeply in [8, 25].

4.3.1.3 Laplacian-based factorization of the averaging matrix with imperfect knowledge of $\bar{x}$

For all methods proposed in previous case 4.3.1.2, the consensus value $\bar{x}$ is the foremost required parameter. We now relax this requirement by means of a novel method that deals with Laplacian eigenvalues estimation when an exact average consensus value is unknown.

In what follows, we assume that the nodes run a standard average consensus protocol (1.2.1), which is stopped after M iterations, and store their local values $x_{i,M}$. In this study, we want to show the performance of the proposed algorithm with respect to some values of M.

Since $\bar{\mathbf{x}}$ is unknown exactly, for a sufficiently large value of M, $\mathbf{x}(M)$ can be viewed as its

rational approximation. Therefore, our aim is to find the coefficients c_k that minimize the quadratic error on the final consensus value:

$$E(\mathbf{c}) = \left\| \sum_{k=0}^{h} c_k \mathbf{q}(k) - \mathbf{x}(M) \right\|^2 . \tag{4.3.36}$$

Then, we can deduce the set $\mathbf{S}_1 = \{\boldsymbol{\alpha}_k\}_{k=1,\cdots,h}$ and that of the inverse of the Laplacian eigenvalues $\mathbf{S}_2 = \{\frac{1}{\lambda_i}, i = 2, \ldots, D\}$, which belongs to $\mathbf{S}_1 : \mathbf{S}_2 \subset \mathbf{S}_1$.

In order to solve the problem in a distributed way, it can be reformulated as follows:

$$\begin{aligned} \min_{\boldsymbol{c}_k \in I\!R^{N\times 1}, k=0,\ldots,h} \quad & \frac{1}{2} \sum_{i=1}^{N} (\sum_{k=0}^{h} c_{i,k} q_{i,k} - \bar{x}_{i,M})^2, \\ \text{subject to} \quad & c_{i,k} = c_{j,k}, \quad i = 1, \ldots, N; j \in N_i; k = 0, \ldots, h. \\ & \mathbf{c}_i = (c_{i,0}, c_{i,1}, \cdots, c_{i,h})^T \in C_i, \end{aligned} \tag{4.3.37}$$

where $c_{i,k}$ is the k^{th} local polynomial coefficient and C_i stands for the constrained set with respect to node i. It is defined as $C_i = \{\mathbf{c}_i \in I\!R^{h+1}, i = 1, \ldots, N : \mathbf{q}_l^T \mathbf{c}_i = \bar{x}_{i,M}$ with $\mathbf{q}_l \in I\!R^{h+1}, \ l \in N_i \cup i\}$ or equivalently as $C_i = \{\mathbf{c}_i \in I\!R^{h+1}, i = 1, \ldots, N : \mathbf{Q}_i \mathbf{c}_i = \bar{\mathbf{x}}_{i,M} \mathbf{1}, \ \mathbf{Q}_i \in I\!R^{d_i+1 \times h+1}\}$ where $\bar{x}_{i,M}$ being the local version of $\bar{x}$ at iteration M and $\mathbf{Q}_i = [\mathbf{q}(0) \ \mathbf{q}(1) \ldots \mathbf{q}(h)]$, being full row rank by construction.

This constraint set allows to enforce not only the local coefficients vector $\mathbf{c}_i$ to be equal in a given neighborhood but also to enforce the scalar product $\mathbf{q}_i^T \mathbf{c}_i$ to be globally equal.

In the literature, there exist several methods dealing with distributed consensus optimization (4.3.37), including distributed descent algorithms [57], dual averaging methods [18], and the ADMM [11, 23]. However, we still make use of ADMM, which has become a popular approach for solving convex minimization problems by means of parallelization.

One can note that by introducing the auxiliary variables $\{\mathbf{z}_{ij}\}$, the following constrained convex optimization problem is absolutely equivalent to problem (4.3.37):

$$\min_{\boldsymbol{c}_k \in I\!R^{N\times 1}, k=0,\ldots,h} \quad \frac{1}{2} \sum_{i=1}^{N} (\sum_{r=0}^{h} c_{i,k} q_{i,k} - \bar{x}_{i,M})^2. \tag{4.3.38}$$

$$\begin{aligned} \text{subject to} \quad & c_{i,k} = z_{ij}^k, \quad i = 1, \ldots, N; j \in N_i \\ & z_{ji}^k = z_{ij}^k, \quad k = 0, \ldots, h. \\ & z_{ij}^k = \text{sign}(k) z_{ij}^k, \\ & \mathbf{c}_i \in C_i, \end{aligned} \tag{4.3.39}$$

where $sign(k) = \begin{cases} 1 & \text{if } k \text{ is even,} \\ -1 & \text{if } k \text{ is odd} \end{cases}$. Since the graph is assumed to be connected, the constraints (4.3.39) force $\mathbf{c}_i = \mathbf{c}_j, j \in N_i$. We can then write the associated augmented

Lagrangian:

$$L_\rho(\mathbf{c},\mathbf{z},\mathbf{y}) \;=\; \sum_{i=1}^{N}\left[\frac{1}{2}\sum_{k=0}^{h}(c_{i,k}q_{i,k}-\bar{x}_{i,M})^2+\sum_{k=0}^{h}\left(\sum_{j\in N_i}y_{ij}^k(c_{i,k}-z_{ij,k})\right)+\sum_{j\in N_i}\frac{\rho}{2}\|\mathbf{c}_i-\mathbf{z}_{ij}\|^2\right].$$

or in vector form:

$$L_\rho(\mathbf{c},\mathbf{z},\mathbf{y}) \;=\; \sum_{i=1}^{N}\left[\frac{1}{2}(\mathbf{q}_i^T\mathbf{c}_i-\bar{x}_{i,M})^2+\sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i-\mathbf{z}_{ij})+\sum_{j\in N_i}\frac{\rho}{2}\|\mathbf{c}_i-\mathbf{z}_{ij}\|^2\right].$$

The ADMM solution to this problem acts in three steps:

- First, minimization with respect to $\mathbf{c}_i$:

$$\mathbf{c}_i[t+1]=\Omega_{C_i}[\text{argmin}\quad L(\mathbf{c}_i,\mathbf{z}_{ij}[t],\mathbf{y}_i[t])]. \tag{4.3.40}$$

  where $\Omega_{C_i}[\cdot]$ stands for the projection in the constraints set of the vector in the argument.

- Second, minimization with respect to $\mathbf{z}_{ij}$ with the constraint $\mathbf{z}_{ji}=\mathbf{z}_{ij}$:

$$\mathbf{z}_{ij}[t+1]=\text{argmin}\quad L(\mathbf{c}_i[t+1],\mathbf{z}_{ij},\mathbf{y}_i[t]). \tag{4.3.41}$$

- Third, Lagrange multiplier update:

$$\mathbf{y}_{ij}[t+1]=\mathbf{y}_{ij}[t]+\rho(\mathbf{c}_i[t+1]-\mathbf{z}_{ij}[t+1]). \tag{4.3.42}$$

Solving the sub-optimization problem (4.3.40) through $\frac{\partial L_\rho(\mathbf{c},\mathbf{z},\mathbf{y})}{\partial \mathbf{c}_i}=0$, we get:

$$(\mathbf{q}_i\mathbf{q}_i^T-\bar{x}_{i,M})\mathbf{q}_i+\sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i-\mathbf{z}_{ij})+\frac{\rho}{2}\sum_{j\in N_i}(\mathbf{c}_i-\mathbf{z}_{ij})^2=0.$$

$$\Leftrightarrow(\mathbf{q}_i\mathbf{q}_i^T+\rho d_i\mathbf{I}_{h+1})\mathbf{c}_i=\bar{x}_{i,M}\mathbf{q}_i+\rho\sum_{j\in N_i}\mathbf{z}_{ij}-\sum_{j\in N_i}\mathbf{y}_{ij}.$$

Then,

$$\hat{\mathbf{c}}_i[t+1]=(\mathbf{q}_i\mathbf{q}_i^T+\rho d_i\mathbf{I}_{h+1})^{-1}(\bar{x}_{i,M}\mathbf{q}_i+\rho\sum_{j\in N_i}\mathbf{z}_{ij}[t]-\sum_{j\in N_i}\mathbf{y}_{ij}[t]).$$

Then, projecting the obtained solution in the constraints set

$$\mathbf{c}_i[t+1]=\Omega_{C_i}[\hat{\mathbf{c}}_i[t+1]],$$

we get:

$$\mathbf{c}_i[t+1]=\tilde{\mathbf{Q}}_i\bar{\mathbf{x}}_{i,M}+(\mathbf{I}_{h+1}-\tilde{\mathbf{Q}}_i\mathbf{Q})\hat{\mathbf{c}}_i[t+1], \tag{4.3.43}$$

with $\tilde{\mathbf{Q}}_i=\mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$ and $\mathbf{Q}_i$ be a full-row rank matrix.

Next, solving the sub-optimization problem (4.3.41) yields

$$\mathbf{z}_{ij}[t+1] = \frac{1}{2}(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + \frac{1}{2\rho}(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]). \tag{4.3.44}$$

Then, we project $\mathbf{z}_{ij}[t+1]$ on the sign constraint as:

$$z_{ij}^k[t+1] = \begin{cases} \max(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) = 0. \\ \min(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) \neq 0. \end{cases} \tag{4.3.45}$$

The corresponding ADMM algorithm is then described as follows:

Algorithm 7 (***ADMM-based distributed estimation of stepsize α_k in the case of inexact $\bar{x}$***)

1. *Inputs:*
   - *Given the number of agents N,*
   - *Define $h = N-1$, the initial input $\{x_i(0)\}$, $i = 1, \cdots, N$, and $\bar{x}_{i,M} = x_i(M)$ for some M,*
   - *Each node forms the vector $\mathbf{q}_i$, and the matrices $\mathbf{Q}_i$ and $\tilde{\mathbf{Q}}_i$.*
2. *Initialization:*
   - *Penalty parameter ρ,*
   - *Random initial values of the coefficients $\{c_{i,k}\}$,$k = 0, \ldots, h$, at each node i, $i = 1, \ldots, N$.*
   - *Random initial values of $\mathbf{z}_{ij}[0]$ and $\mathbf{y}_{ij}[0]$.*
   - *Set $t = 0$;*
3. *Update Process:*
   (a) *Set $t := t + 1$;*
   At each iteration, node i sends a message including its local coefficient $c_{i,k}$. After receiving the information from its neighbors, each node carries out the update process.
   (b) *Update equations:*
   - *Compute $\mathbf{c}_i[t+1]$ using (4.3.43).*
   - *Compute $\mathbf{z}_{ij}[t+1]$ using (4.3.44), (4.3.45).*
   - *Update the Lagrange multipliers $\mathbf{y}_{ij}[t+1]$ using (4.3.42).*

   (c) *Return to 3a or stop the iterations if a stopping criterion is reached.*
4. *Each agent builds a polynomial p_i with the set of coefficients $c_{i,k}[t+1]$.*

5. *Each agent computes the set of the stepsizes* $\{\alpha_{i,k}\}$ *as the roots of the polynomial* p_i.

When the coefficients c_k are approximately obtained, we can make use of Algorithms 9 and 10 that will be presented later in Subsection 4.3.2 to estimate the whole spectrum of Laplacian matrix $\mathbf{L}$, that can be used to estimate the network robustness indices.

4.3.1.4 Laplacian-based factorization of the averaging matrix with unknown consensus value $\bar{x}$

According to the previous section, one way for estimating the Laplacian eigenvalues is:

- solve the average consensus problem.
- solve the averaging matrix factorization problem (4.3.15), Algorithms 5 and 6.
- retrieve the Laplacian eigenvalues using Algorithm 9 and 10.

Instead of following these steps, in this section, we propose a method to merge the first two steps. For this purpose, we propose to solve a convex combination of two convex functions, i.e.

$$\begin{aligned} \min_{\bar{x}_i, \mathbf{c}_i, i=1,\cdots,N} \quad & \frac{\theta}{2} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 + \frac{1-\theta}{2} \sum_{i=1}^{N} (\bar{x}_i - x_i(0))^2, \\ \text{s.t.} \quad & \mathbf{Q}_i \mathbf{c}_i = \bar{x}_i \mathbf{1}, \quad i = 1, \ldots, N, \end{aligned} \tag{4.3.46}$$

with θ- an adjustable scalar parameter ($0 \leq \theta \leq 1$), $\bar{x}_i$- a local estimate of the average value, and $x_i(0)$- a fixed initial value of node i.

It can be seen that, the objective function (4.3.46) is a combination of two convex functions. The first one takes into account the estimation of the polynomial coefficients whose roots are the inverse of the nonzero distinct Laplacian eigenvalues while the second one concerns the average consensus problem. And this convex optimization problem can be solved efficiently using the ADMM approach.

In order to derive an ADMM algorithm for the problem (4.3.46), we introduce the auxiliary variables $\{\mathbf{z}_{ij}, \mu_{ij}\}$. Therefore, the following constrained convex optimization problem is absolutely equivalent to the problem (4.3.46):

$$\begin{aligned} \min_{\bar{x}_i, \mathbf{c}_i, i=1,\cdots,N} \quad & \frac{\theta}{2} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 + \frac{1-\theta}{2} \sum_{i=1}^{N} (\bar{x}_i - x_i(0))^2, \\ \text{s.t} \quad & \bar{x}_i = \mu_{ij}, \quad i = 1, \ldots, N, \quad j \in N_i. \end{aligned}$$

$$\begin{aligned}&\mathbf{c}_i = \mathbf{z}_{ij}.\\&z_{ij}^k = sign(k) z_{ij}^k,\\&\mathbf{c}_i \in C_i(\bar{x}_i).\end{aligned}$$

where

- $sign(k) = \begin{cases} 1 & \text{if } k \text{ is even,} \\ -1 & \text{if } k \text{ is odd} \end{cases}$.
- $C_i(\bar{x}_i)$ stands for the constrained set with respect to node i. It is defined as $C_i = \{\mathbf{c} \in I\!R^{h+1} |\ \mathbf{Q}_i\mathbf{c} = \bar{x}_i\mathbf{1}\}$.

Now, introducing the dual variables $\boldsymbol{\upsilon}$ and $\mathbf{y}$, the augmented Lagrangian is defined as :

$$L_{\rho_1\rho_2}(\bar{\mathbf{x}}, \mathbf{c}, \boldsymbol{\mu}, \mathbf{z}, \boldsymbol{\upsilon}, \mathbf{y}) = L_{\rho_1}(\mathbf{c}, \mathbf{z}, \mathbf{y}) + L_{\rho_2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon}).$$

where:

$$\begin{aligned}L_{\rho_1}(\mathbf{c}, \mathbf{z}, \mathbf{y}) &= \frac{\theta}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i - \mathbf{c}_j\|^2 + \sum_{j\in N_i}\frac{\rho_1}{2}\|\mathbf{c}_i - \mathbf{z}_{ij}\|^2 + \sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i - \mathbf{z}_{ij}),\\ L_{\rho_2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon}) &= \frac{1-\theta}{2}\sum_{i=1}^{N}(\bar{x}_i - x_i(0))^2 + \sum_{j\in N_i}\frac{\rho_2}{2}(\bar{x}_i - \mu_{ij})^2 + \sum_{j\in N_i}\upsilon_{ij}(\bar{x}_i - \mu_{ij}),\end{aligned}$$

where ρ_1, ρ_2 are penalty terms assumed to be constant.

ADMM provides an iterative solution that acts following the five steps below:

- Minimization with respect to the average consensus value $\bar{x}_i$:

$$\bar{x}_i[t+1] = \arg\min\ L_{\rho_2}(\bar{x}_i[t], \mu_{ij}[t], \upsilon_{ij}[t]). \tag{4.3.47}$$

- Minimization with respect to the polynomial coefficients $\mathbf{c}_i$:

$$\hat{\mathbf{c}}_i = \text{argmin}\, L_{\rho_1}(\mathbf{c}_i[t], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t]), \tag{4.3.48}$$

$$\mathbf{c}_i[t+1] = \Omega_{C_i(\bar{x}_i[t+1])}[\hat{\mathbf{c}}_i] \tag{4.3.49}$$

  where $\Omega_{C_i(\bar{x}_i[t+1])}[\cdot]$ stands for the projection into the constraints set $C_i(\bar{x}_i[t+1])$ of the vector in the argument.

- Minimization with respect to the auxiliary variables μ_{ij} with the constraint $\mu_{ji} = \mu_{ij}$:

$$\mu_{ij}[t+1] = \text{argmin}\, L_{\rho_2}(\bar{x}_i[t+1], \mu_{ij}[t], \upsilon_{ij}[t]). \tag{4.3.50}$$

- Minimization with respect to the auxiliary variables $\mathbf{z}_{ij}$:

$$\mathbf{z}_{ij}[t+1] = \text{argmin } L_{\rho_1}(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t]). \tag{4.3.51}$$

- Lagrange multipliers update:

$$\upsilon_{ij}[t+1] = \upsilon_{ij}[t] + \rho_2(\bar{x}_i[t+1] - \mu_{ij}[t+1]), \tag{4.3.52}$$

$$\mathbf{y}_{ij}[t+1] = \mathbf{y}_{ij}[t] + \rho_1(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]). \tag{4.3.53}$$

Solving the sub-optimization problems (4.3.47) and (4.3.48) is equivalent to solve the following equations:

1. $\partial \frac{L_{\rho_1}(\mathbf{c},\mathbf{z},\mathbf{y})}{\partial \mathbf{c}_i} = 0$

$$\frac{\partial L_{\rho_1}(\mathbf{c},\mathbf{z},\mathbf{y})}{\partial \mathbf{c}_i} = \theta \sum_{j\in N_i}(\mathbf{c}_i - \mathbf{c}_j) + \sum_{j\in N_i}\mathbf{y}_{ij} + \sum_{j\in N_i}\rho_1(\mathbf{c}_i - \mathbf{z}_{ij}) = 0.$$

We get:

$$\Leftrightarrow \hat{\mathbf{c}}_i[t+1] = \frac{\theta\sum_{j\in N_i}\mathbf{c}_j[t] + \rho_1\sum_{j\in N_i}\mathbf{z}_{ij}[t] - \sum_{j\in N_i}\mathbf{y}_{ij}[t]}{d_i(\theta+\rho_1)}.$$

2. $\partial \frac{L_{\rho_2}(\bar{\mathbf{x}},\boldsymbol{\mu},\boldsymbol{\upsilon})}{\partial \bar{x}_i} = 0$

$$\frac{\partial L_{\rho_2}(\bar{\mathbf{x}},\boldsymbol{\mu},\boldsymbol{\upsilon})}{\partial \bar{x}_i} = (1-\theta)(\bar{x}_i - x_i(0)) + \sum_{j\in N_i}\upsilon_{ij} + \sum_{j\in N_i}\rho_2(\bar{x}_i - \mu_{ij}) = 0$$

$$\begin{aligned}
&\Leftrightarrow \quad (1-\theta+\rho_2 d_i)\bar{x}_i = (1-\theta)x_i(0) + \rho\sum_{j\in N_i}\mu_{ij} - \sum_{j\in N_i}\upsilon_{ij} \\
&\Leftrightarrow \quad \bar{x}_i[t+1] = \frac{(1-\theta)x_i(0) + \rho_2\sum_{j\in N_i}\mu_{ij}[t] - \sum_{j\in N_i}\upsilon_{ij}[t]}{1-\theta+\rho_2 d_i}.
\end{aligned} \tag{4.3.54}$$

Then, projecting the obtained solution in the constraints set

$$\mathbf{c}_i[t+1] = \Omega_{C_i(\bar{x}_i[t+1])}[\hat{\mathbf{c}}_i[t+1]],$$

we get:

$$\mathbf{c}_i[t+1] = \tilde{\mathbf{Q}}_i\bar{x}_i[t+1]\mathbf{1} + (\mathbf{I}_{h+1} - \tilde{\mathbf{Q}}_i\mathbf{Q})\hat{\mathbf{c}}_i[t+1], \tag{4.3.55}$$

with $\tilde{\mathbf{Q}}_i = \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$, $\mathbf{Q}_i$ being full row rank by construction.

Next, solving the sub-optimization problems (4.3.50) and (4.3.51), in the same way, yields

1. The derivative of $\mathbf{L}_{\rho 2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon})$ with respect to μ_{ij} with respect to the constraint $\mu_{ij} = \mu_{ji}$:

$$\begin{aligned} &\frac{\partial L_{\rho 2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon})}{\partial \mu_{ij}} = -\upsilon_{ij} - \upsilon_{ji} - \rho_2(\bar{x}_i + x_j) + 2\rho_2\mu_{ij} = \mathbf{0} \\ &\Leftrightarrow \mu_{ij}[t+1] = \frac{\bar{x}_i[t+1] + \bar{x}_j[t+1]}{2} + \frac{\upsilon_{ij}[t] + \upsilon_{ji}[t]}{2\rho_2}. \end{aligned} \tag{4.3.56}$$

2. The derivative of $\mathbf{L}_{\rho_1}(\mathbf{c}, \mathbf{z}, \mathbf{y})$ with respect to $\mathbf{z}_{ij}$

$$\begin{aligned} &\partial \tfrac{L_{\rho 1}(\mathbf{c}_i, \mathbf{z}_{ij}, \mathbf{y}_{ij})}{\mathbf{z}_{ij}} = \mathbf{0} \\ \Leftrightarrow \quad &-\mathbf{y}_{ij}[t] - \rho(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]) = 0 \\ \Leftrightarrow \quad &\mathbf{z}_{ij}[t+1] = \mathbf{c}_i[t+1] + \tfrac{1}{\rho_1}\mathbf{y}_{ij}[t]. \end{aligned} \tag{4.3.57}$$

Then, we project $\mathbf{z}_{ij}[t+1]$ on the sign constraint as:

$$z_{ij}^k[t+1] = \begin{cases} \max(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) = 0. \\ \min(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) \neq 0. \end{cases} \tag{4.3.58}$$

The parameter θ allows weighting the two combined cost functions. Since the constraint set $C_i(\bar{x}_i)$ depends on the consensus value $\bar{x}_i$. The optimization procedure can initially give more weight to the average consensus problem and then to the factorization one. For this purpose, instead of keeping θ constant, we adopt a time varying coefficient:

$$\theta(t) = \frac{1 - e^{-\beta t}}{1 + e^{-\beta t}}, \quad 0 < \beta < 1, \tag{4.3.59}$$

to keep it in the range $[0, 1]$ and to increase the importance of the factorization progressively while decreasing that of the average consensus.

Remark 12

Practically, we can see that the average consensus term converges much faster than the term relative to the averaging matrix factorization problem. Hence, θ can be chosen close to 1.

The corresponding ADMM algorithm is then described as follows:

Algorithm 8 (*ADMM-based joint distributed average consensus and estimation of Laplacian eigenvalues*)

1. *Inputs:*
   - *Given the number of nodes N, $h = N - 1$,*
   - *The initial measurements values $\{x_i(0)\}$, $i = 1, \cdots, N$,*
2. *Initialization:*

- *Penalty parameters* ρ_1, ρ_2, *and* β.
- *Random initial values of the coefficients* $\{c_{i,k}\}$,$k = 0, \ldots, h$, *at each node* i, $i = 1, \ldots, N$.
- *Random initial values of* $\mathbf{z}_{ij}, \boldsymbol{y}_{ij}$ *and* μ_{ij}, υ_{ij}.
- *Set* $t = 0$;

3. *Update Process:*

   (a) *Set* $t := t + 1$;
   At each iteration, node i *sends a message including its local value* $\bar{x}_i$, *and local coefficient* $c_{i,k}$.
   After receiving the information from its neighbors, each node i *carries out the update process.*

   (b) *Update equations:*
   - *Compute* $\bar{x}_i[t+1]$ *using (4.3.54)*
   - *Compute* $\mathbf{c}_i[t+1]$ *using (4.3.55).*
   - *Compute* $\mu_{ij}[t+1]$ *using (4.3.56).*
   - *Compute* $\mathbf{z}_{ij}[t+1]$ *using (4.3.57), (4.3.58).*
   - *Update the Lagrange multipliers* $\upsilon_{ij}, \boldsymbol{y}_{ij}[t+1]$ *using (4.3.52), (4.3.53).*
   - *Update* $\theta(t)$ *using (4.3.59).*

   (c) *Return to 3a or stop the iterations if a stopping criterion is reached.*

4. *Each agent build a polynomial* P_i *with the set of coefficients* $\mathbf{c}_i[t+1]$.

5. *Each agent compute the set of the stepsizes* $\{\alpha_{i,k}\}$ *as the roots of the polynomial* P_i, *the set of Laplacian eigenvalues using Algorithm 1.*

6. *The average consensus value is* $\bar{x}_i[t+1]$.

4.3.2 Laplacian spectrum retrieving

At each node, the set $\mathbf{S}_1$ of step-sizes obtained by means of the proposed Algorithms (Algorithms 4, 5, 6, 7, 8) contains the set $\mathbf{S}_2$ of the inverse of nonzero distinct Laplacian eigenvalues.

Let $\hat{\bar{x}}_i$ be the final consensus value reconstructed by $\hat{\bar{x}}_i = \sum_{k=0}^{h} c_k q_{i,k} = \mathcal{P}_i(\mathbf{c}, \mathbf{q}_i)$ with the coefficients obtained by means of Algorithms 4, 5, 6, 7 and 8.

Following the Proposition 3, the idea is to reduce, step-by-step, the degree of the polynomial $\mathcal{P}_i(\mathbf{c}, \mathbf{q}_i)$ by removing one element of $\mathbf{S}_1$ at each step. We know that if the removed element is also an element of $\mathbf{S}_2$ then $\hat{\tilde{x}}_i \neq \tilde{\mathcal{P}}_i(\mathbf{c}, \tilde{\mathbf{q}}_i)$ where $\tilde{\mathcal{P}}_i$ is the polynomial with reduced degree with respect to node i. Algorithm 9 describes the proposed procedure.

Algorithm 9 (*Laplacian eigenvalues retrieving*)

Given the set of step size $\boldsymbol{S}_1$, *the final consensus value* $\hat{\tilde{x}}_i$, *the measurements vector* $\mathbf{q}_i$ *and the threshold* ϵ

1. *Initialization:* $\boldsymbol{S}_2 = \emptyset$ *and* $\boldsymbol{S}_3 = \boldsymbol{S}_1$.
2. *while* $\boldsymbol{S}_3 \neq \emptyset$, *select an element* α_j *of* $\boldsymbol{S}_3$.
3. *From the set* $\boldsymbol{S}_3 \backslash \{\alpha_j\}$ *compute the coefficients* c_i *using (4.3.5) and then form* $\tilde{\mathbf{c}}_i$.
4. *If* $|\hat{\tilde{x}}_i - \tilde{\mathbf{c}}^T \mathbf{q}_i| > \epsilon$, *include* α_j *in* $\boldsymbol{S}_2$ *so that* $\boldsymbol{S}_2 := \boldsymbol{S}_2 \bigcup \{\alpha_j\}$, *set* $\boldsymbol{S}_3 := \boldsymbol{S}_3 / \{\alpha_j\}$, *and return to 2.*
5. *If* $|\hat{\tilde{x}}_i - \tilde{\mathbf{c}}^T \mathbf{q}_i| < \epsilon$, *set* $\boldsymbol{S}_3 := \boldsymbol{S}_3 / \{\alpha_j\}$, *and return to 2.*
6. *If* $\boldsymbol{S}_3 = \emptyset$, *deduce the Laplacian eigenvalues as the inverses of the elements in* $\boldsymbol{S}_2$.

We can note that this algorithm is to be run strictly locally.

As a result, we obtain the set of nonzero distinct Laplacian eigenvalues $\Lambda = \{\lambda_2, \ldots, \lambda_{D+1}\}$. In order to obtain the whole spectrum $sp(\mathbf{L}) = \{\lambda_1^{m_1}, \lambda_2^{m_2}, \ldots, \lambda_{D+1}^{m_{D+1}}\}$, the multiplicities m_i should be defined. Since all of them are integers, the problem now resorts to an integer programming problem.

Given the set of nonzero distinct Laplacian eigenvalues Λ, d_i the degree of each node i, inspired from an interesting property $\sum_{i=1}^{N} \lambda_i(\mathbf{L}) = \sum_{i=1}^{N} d_i$ (see in Section 2.3), the optimization is now described as follows:

$$\begin{aligned} \min_{\mathbf{m} \in \mathbb{N}^D} \quad & \Lambda^T \mathbf{m} - \sum_{i=1}^{N} d_i, \\ \text{subject to} \quad & \sum_{j=1}^{D} m_j = N - 1, \ j = 1, \ldots, D. \\ & m_j \geq 1, \text{integer}. \end{aligned} \tag{4.3.60}$$

where $D = |\Lambda|$ and $\mathbf{m}$ is the vector of eigenvalues multiplicities.

In the literature, the branch and bound method is the basic workhorse technique for solving integer programming problems. The linear programming (LP) relaxation (with no

integer restrictions) is adopted to estimate the optimal solution of an integer programming [13]. Therefore, problem (4.3.60) is now rewritten strictly equivalently in linear programming form as follows:

$$\begin{aligned} \min_{\mathbf{m}\in\mathbb{N}^D} \quad & \Lambda^T\mathbf{m}, \\ \text{subject to} \quad & \sum_{j=1}^{D} m_j = N-1; \\ & \Lambda^T\mathbf{m} = \sum_{i=1}^{N} d_i, \qquad (\mathbf{M}) \\ & m_j \geq 1, \end{aligned} \tag{4.3.61}$$

where $\mathbf{M}$ is the constraint set without integer restriction. Hence, $\mathbf{M}$ can be expressed as $\mathbf{M} = \{\mathbf{m} \in I\!R^D | \sum_{j=1}^{D} m_j = N-1, \Lambda^T\mathbf{m} = \sum_{i=1}^{N} d_i \text{ and } m \geq 1\}$. The problem (4.3.61) without integer restriction is called linear programming (LP) relaxation.

The branch-and-bound-based algorithm is described as:

Algorithm 10 (*Brand-and-bound-based multiplicities estimation*)

1. *Initialization:*
   (a) *Number of nodes* N, d_i, Λ.
   (b) $t = 0$;
   (c) $\boldsymbol{S} = \emptyset$ *set of integer solutions.*
   (d) $F_{best} = \emptyset$ *function value of feasible solution.*
   (e) $V = \emptyset$ *set of nodes, that the program will visit.*
   (f) *Compute* $\mathbf{m}[0] = argmin\ \Lambda^T\mathbf{m}$, *s.t* $\boldsymbol{m} \in \boldsymbol{M}$
      - *If the stopping criteria is satisfied, then stop.*
      - *Compute the upper bound (UB), which is the function value of the rounded-up feasible solution,* $F_{best} = UB$.
      - $\boldsymbol{S}_4 = \boldsymbol{m}[0]$.
   (g) *Order of variables in* $\boldsymbol{S}$ *to be branched on.*
2. *While* $\boldsymbol{S}_4 = \emptyset$, *select an ordered variable* $\hat{m}_j$ *to be branched on and* $\boldsymbol{S}_4 := \boldsymbol{S}_4 \backslash \{\hat{m}_j\}$.
   (a) $t = t + 1$;
   (b) *Create 2 new nodes for 2 new sub-problems.*
   (c) $\mathbf{m}^1 = argmin\ \Lambda^T\mathbf{m}$, *s.t* $\boldsymbol{m} \in \boldsymbol{M}_1 = \boldsymbol{M} \cup \{m_j \leq \lfloor \hat{m}_j \rfloor\}$.

(d) $\mathbf{m}^2 = argmin\ \Lambda^T \mathbf{m},\ s.t\ \mathbf{m} \in \boldsymbol{M}_2 = \boldsymbol{M} \cup \{m_j \geq \lceil \hat{m}_j \rceil\}$

- *If* $\mathbf{m}^1$ *(or* $\mathbf{m}^2$*) is integer,* $\boldsymbol{S} = \mathbf{m}^1$*(or* $\mathbf{m}^2$*) then stop.*
- *If there is an infeasible solution, then prun the corresponding node.*
- *Compute* UB_1 *(or* UB_2*).*

(e) *If* $UB_1(UB_2) < UB_2(UB_1)$*, then pick* $\mathbf{m}^1$ *(or* $\mathbf{m}^2$*) is the next node to create 2 new nodes and* $V = V \cup \mathbf{m}^2(\mathbf{m}^1)$*, and* $\boldsymbol{M} = \boldsymbol{M}_1(\boldsymbol{M}_2)$.

3. *Return to step 2a.*

The policy of selecting a variable to be branched on in this algorithm is that the variable with the greatest fractional part is the first order then the variable with the smallest fractional part is the next. After that, the policy is repeated for the rest of unselected variables.

Since, the Laplacian spectrum $sp(\mathbf{L})$ is obtained, we can compute the effective graph resistance $\mathcal{R}$ and the number of spanning trees ξ to assess the robustness of a given network using (1.3.2) and (1.3.1).

4.4 Simulation results

In this section, several examples will be carried out to test the efficiency of the proposed methods to assess the network robustness (via $\mathcal{R}, \xi$). As presented obviously in previous Section, the procedure of all proposed methods are devided into three categories according to the consensus value $\bar{x}$ that are: perfect consensus value (Algorithms 4, 5 and 6); imperfect consensus value (Algorithm 7) and unknown consensus value (Algorithm 8).

In order to evaluate the network robustness, let us take two networks which have the same nodes but different number of edges as examples.

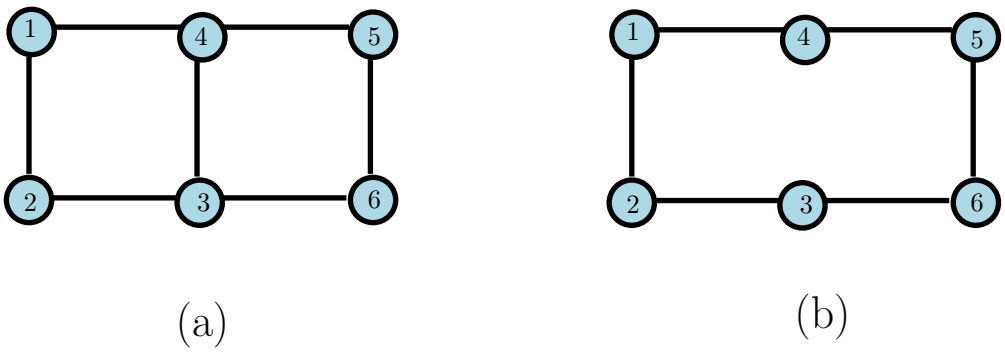


Figure 4.4: Two graphs with the same number of nodes but different number of edges.

The corresponding Laplacian matrices are given by:

$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix} \qquad \mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

with

$sp(\mathbf{L})$	$\{0, 1, 2, 3, 3, 5\}$
$\mathcal{R}$	14.2
ξ	15

with

$sp(\mathbf{L})$	$\{0, 1, 1, 3, 3, 4\}$
$\mathcal{R}$	17.5
ξ	6

4.4.1 Cases of perfect consensus value $\bar{x}$

4.4.1.1 Example 1 (Figure 4.4 (a))

Consider the graph in Figure 4.4(a) with initial vector of state:

$$\mathbf{x}(0) = [0.5832, 0.7400, 0.2348, 0.7350, 0.9706, 0.8669]^T.$$

Assume that the final average consensus value $\bar{x} = 0.6884$ is obtained perfectly by Algorithm 1.

Applying Algorithms proposed in Subsection 4.3.1.2, we can see the difference between them via their performances.

1. Lagrange multiplier method for distributed averaging matrix factorization

We make use of Algorithms 4, 9 to find the stepsizes $\alpha_k = \{1, 0.5, 0.3333, 0.2\}$ and deduce the nonzero distinct eigenvalues $\{1, 2, 3, 5\}$. Furthermore, by using Algorithm 10, the multiplicities of these nonzero distinct Laplacian eigenvalues are achieved to get the whole Laplacian spectrum $sp(\mathbf{L})$.

In practice, the parameters β, μ are chosen to ensure the convergence of the Algorithm 4: $\beta = 0.0222, \mu = 0.2$. After running Algorithm 4, we obtain the set of stepsizes $\{0.9993, 0.2, 0.5, 0.0895, 0.3333\}$. The trajectory of these stepsizes is illustrated in Figure 4.5.

Here, running Algorithm 9, the real set of the nonzero distinct Laplacian eigenvalues can

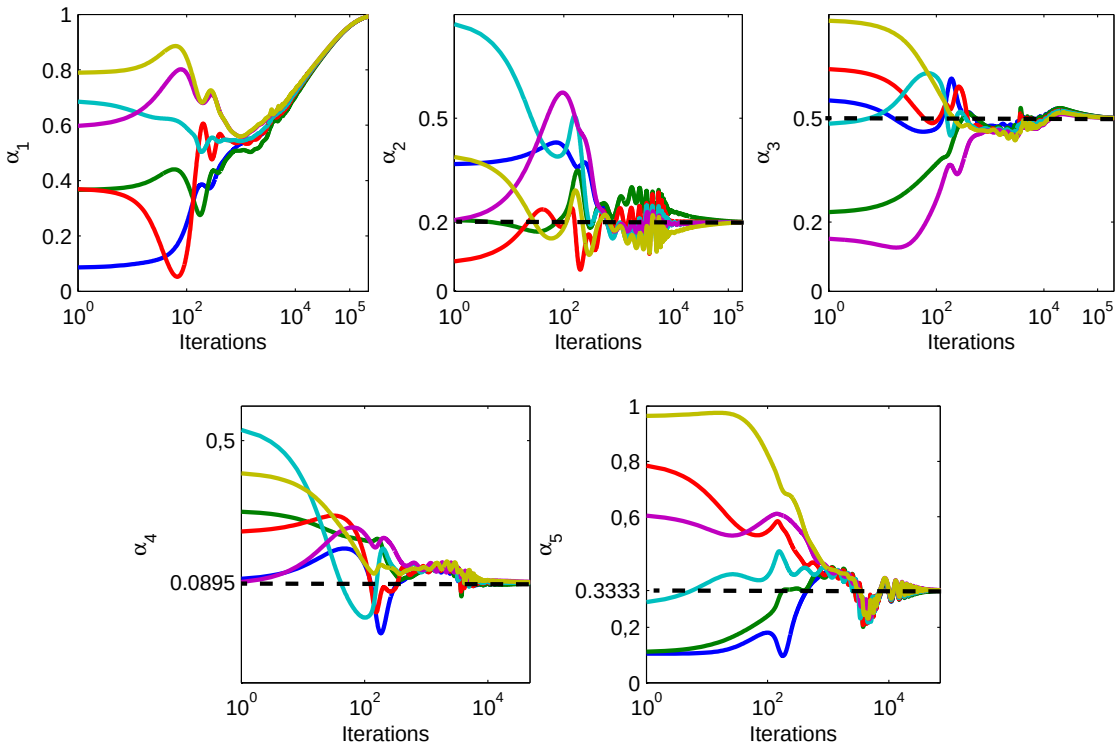


Figure 4.5: Trajectory of the stepsizes α_k.

be inferred $\Lambda = \{1, 5, 2, 3\}$. Finally, by Algorithm 10, the corresponding multiplicities are obtained $\mathbf{m} = \{1, 1, 1, 2\}$. Therefore, we get the whole Laplacian spectrum $sp(\mathbf{L})$.

Due to the fact that the proposed method is based on the gradient concept of a non-convex optimization problem, the convergence speed is obviously slow. Besides that, the choice of the parameters β, μ plays an important role. Figure 4.6 shows the convergence of the Algorithm 4 in this example.

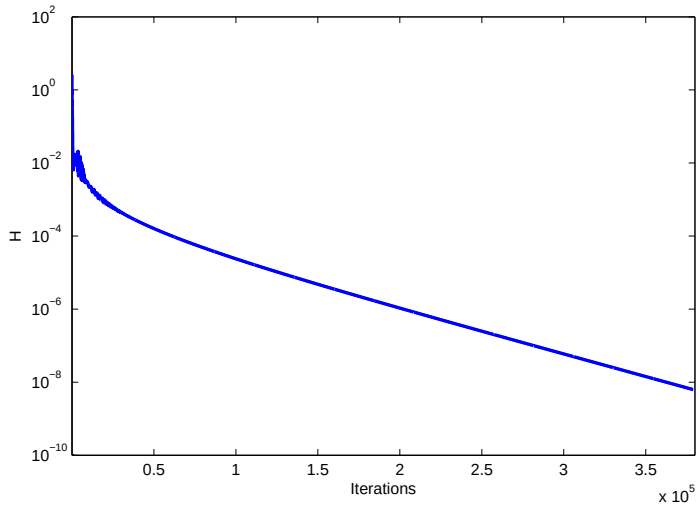


Figure 4.6: Convergence of the Algorithm decribed via the cost function $H(\alpha)$.

By using the equations (1.3.1) and (1.3.2), the estimated robustness indices can be obtained:

- Number of spanning trees $\hat{\xi}$:

$$\hat{\xi} = \frac{1}{N} \prod_{i=2}^{N} \lambda_i(\mathbf{L}) = 15.0135.$$

- Effective graph resistance $\hat{\mathcal{R}}$

$$\hat{\mathcal{R}} = N \sum_{i=2}^{N} \frac{1}{\lambda_i(\mathbf{L})} = 14.1954.$$

2. Distributed projected subgradient method

This convex optimization is implemented with respect to the coefficients $c_k, k = 1, \ldots, 6$. Therefore, the performance is evaluated via the mean square error $(E(c))$ defined as:

$$E(c) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{h} \sum_{j \in N_i} (c_{i,k} - c_{j,k})^2.$$

We now try to find the coefficients $c_k, k = 0, \ldots, 6$ by using the Algorithm 5 and 6, which deduces the stepsizes that are the inverses of the nonzero distinct Laplacian eigenvalues. Then, via Algorithms 9 and 10, the whole spectrum of the Laplacian matrix is obtained.

Figure 4.7 shows the trajectory of the coefficients c_k obtained by means of Algorithm 5.

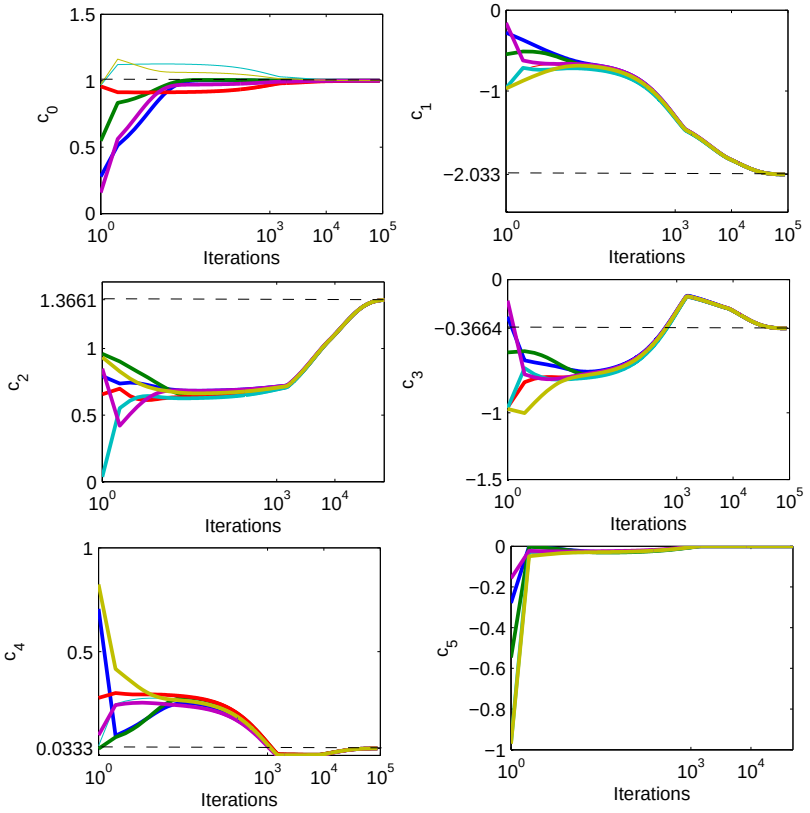


Figure 4.7: Trajectory of the estimated coefficients c_k at each node obtained by means of distributed projected subgradient method

As can be seen that the coefficients $\mathbf{c} = [1, -2.0330, 1.3661, -0.3664, 0.0333, 0]^T$. The convergence is performed in Figure 4.8.

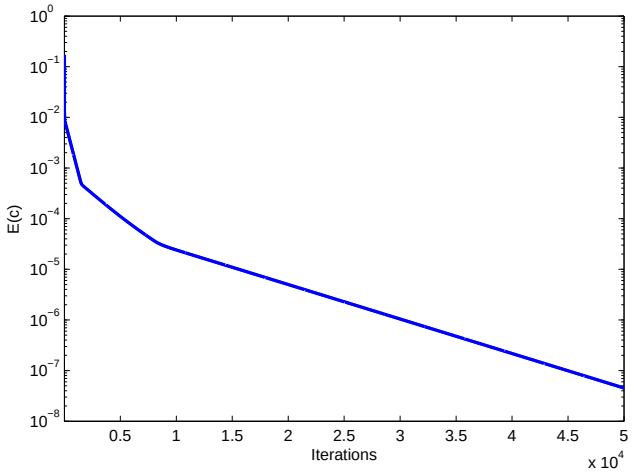


Figure 4.8: Convergence of the distributed projected subgradient method via $E(c)$

In comparison with the Lagrange-based method (Algorithm 4), the speed of convergence is improved.

In this implementation, we have constrained the sign of the elements of the coefficient vector $\mathbf{c}$. Therefore, by $c_{i,5}, i = 1, \ldots, N$ is set to be 0 due to the fact that $c_{i,5} > 0, i = 1, \ldots, N$. Furthermore, by solving the roots of the polynomial of coefficients c_k, we obtain the stepsizes $\alpha_k = \{0.9998, 0.4996, 0.3342, 0.1993\}$ that are definitely close to $\{1, 0.5, 0.3333, 0.2\}$. By making use of Algorithm 10, the whole Laplacian spectrum $sp(\mathbf{L}) = \{1, 2, 3, 3, 5\}$ is obtained.

3. ADMM-based method

Figures 4.9 shows the trajectory of the coefficients c_k obtained corresponding to the value of penalty parameter $\rho = 0.003$ by means of Algorithms 6 .

The obtained coefficients are $\mathbf{c} = [1, -7.0065, 11.4787, -7.1633, 1.8568, -0.1658^T]$ and then $\mathbf{S}_1 = \{4.9732, 1, 0.5, 0.3333, 0.2\}$.

Using Algorithm 9, we get $\mathbf{S}_2 = \{1, 0.5, 0.3333, 0.2\}$ from which we deduce $\Lambda = \{1, 2, 3, 5\}$.

Through Algorithm 10, the whole spectrum $sp(\mathbf{L})$ is achieved.

Another interesting note is the performance of the proposed algorithm with respect to the penalty parameter ρ. Hereafter, we describe a comparison with different choices of ρ. Figure 4.10 illustrates the difference in convergence rate corresponding to different values of penalty parameter ρ

In fact, the main point is that for the distributed projected subgradient method, which is still based on gradient method, the convergence depends on the choice of the stepsize γ.

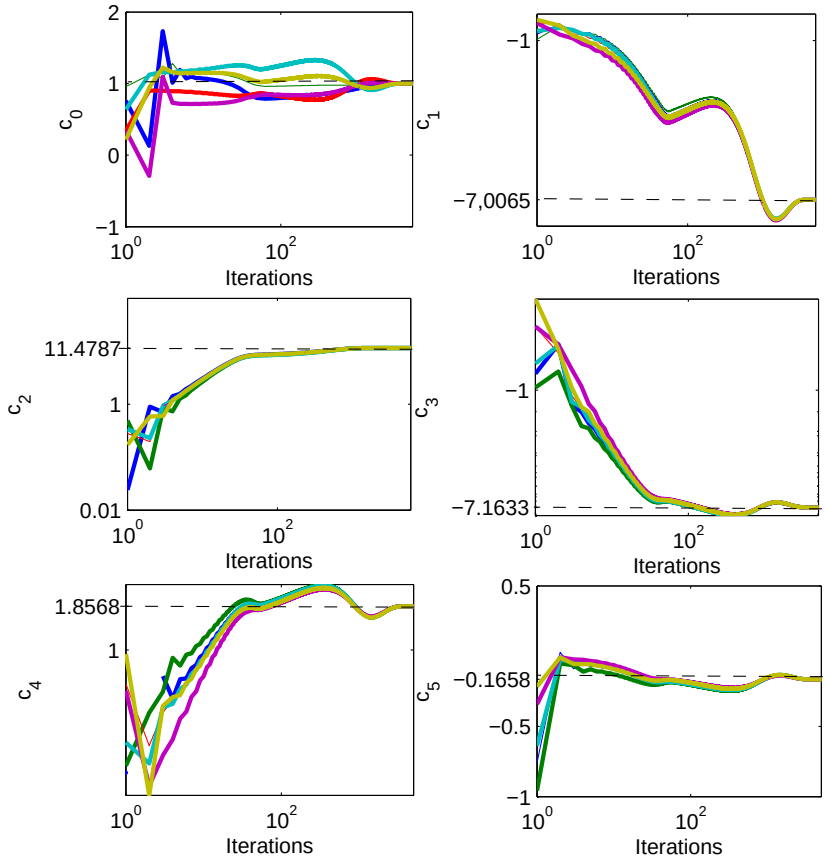


Figure 4.9: Trajectory of the estimated coefficients c_k at each node obtained by means of ADMM

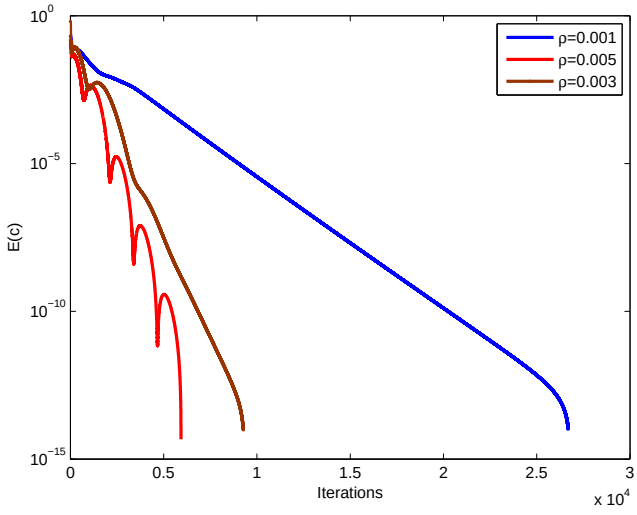


Figure 4.10: Convergence rate of the ADMM-based method with different values of ρ.

On the other hand, the ADMM-based method seems to be the most attracting method when the only parameter needs to be noticed is the penalty parameter ρ to regulate the

speed of convergence.

Figure 4.10 shows that when ρ increases, the speed of convergence is faster. But, the error oscilates as well. However, we still obtain the set of the inverses of the Laplacian eigenvalues.

Since both methods (distributed projected subgradient method and ADMM-based method) obtain the exact Laplacian spectrum $sp(\mathbf{L})$, we can deduce the estimated robustness indices by using equations (1.3.1) and (1.3.2):

- Number of spanning trees ξ:

$$\hat{\xi} = \frac{1}{N}\prod_{i=2}^{N}\lambda_i(\mathbf{L}) = 15. \tag{4.4.1}$$

- Effective graph resistance $\mathcal{R}$:

$$\hat{\mathcal{R}} = \sum_{1\leq i,j\leq N}\mathcal{R}_{ij} = N\sum_{i=2}^{N}\frac{1}{\lambda_i(\mathbf{L})} = 14.2 \tag{4.4.2}$$

The last thing can be noted here is that the solution of the coefficients c_k is not unique since the number of distinct Laplacian eigenvalues ($D = 4$) is less than $h = N - 1 = 5$.

4.4.1.2 Example 2 (Figure 4.4 (b))

The considered network is modelled by a 6-node cycle graph in Figure 4.4(b) with the initial values for the consensus protocol:

$$\mathbf{x}(0) = [0.3074,\ 0.4561,\ 0.1017,\ 0.9954,\ 0.3321,\ 0.2973]^T.$$

Moreover, we know that the number of distinct nonzero Laplacian eigenvalue D is equal to the diameter of this graph ($d(G) = 3$). Therefore, we set $h = D = 3$ and start to run the algorithm to assess the network robustness.

1. Lagrange multiplier method for distributed averaging matrix factorization

Picking $\mu = 0.15$, $\beta = 0.003$, and running Algorithm 4, we obtain 3 distinct inverses of the Laplacian eigenvalues $\Lambda = \{0.3333, 1, 0.25\}$. Then, by applying the Algorithm 10, the corresponding multiplicities are estimated $\mathbf{m} = \{2, 2, 1\}$.

Figure 4.11 shows the trajectory of each node on the stepsize $\alpha_k, k = 1, \ldots, 3$.

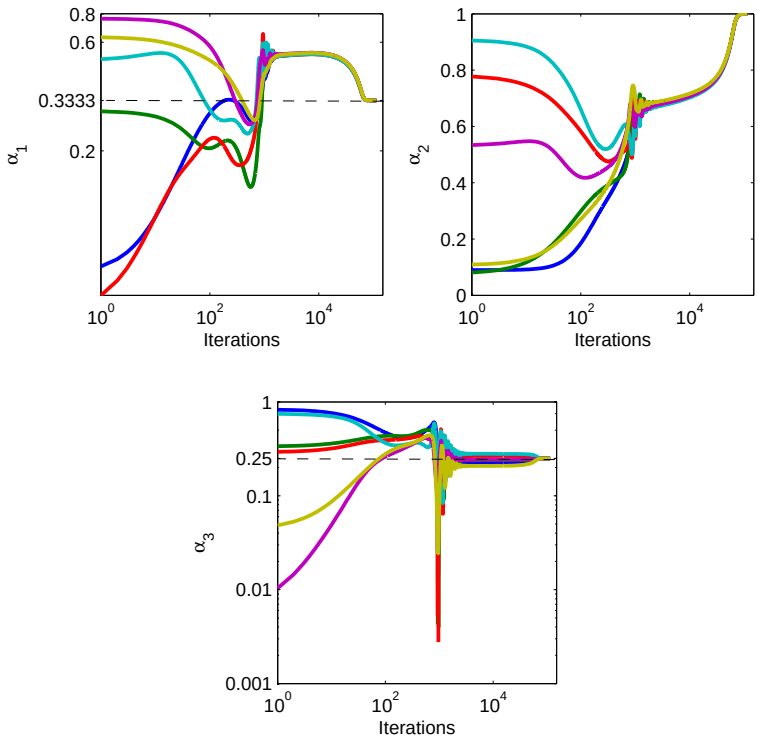


Figure 4.11: Zoom in the trajectory of each node on the stepsize α_k

Figure 4.12 shows the convergence of Algorithm 4 in this example.

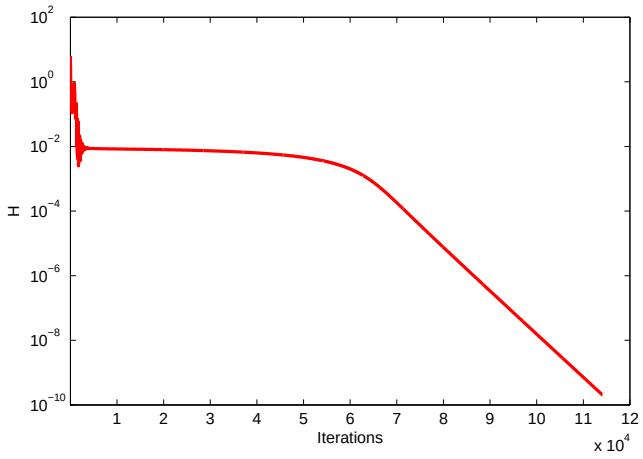


Figure 4.12: Convergence of Algorithm 4 performed through the cost function $H(\alpha)$

2. Distributed projected subgradient method

Figure 4.13 shows the trajectory of the coefficients c_k obtained by means of Algorithms 5.

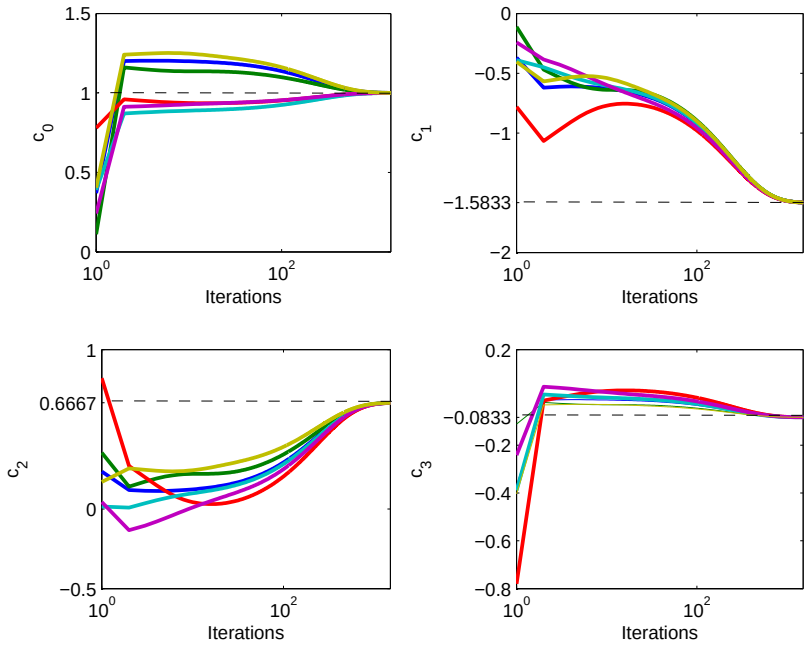


Figure 4.13: Trajectory of the estimated coefficients c_k at each node obtained by means of distributed projected subgradient method

As can be seen that the coefficients $\mathbf{c} = [1, -1.5833, 0.6667, -0.0833]^T$. The convergence is performed in Figure 4.14.

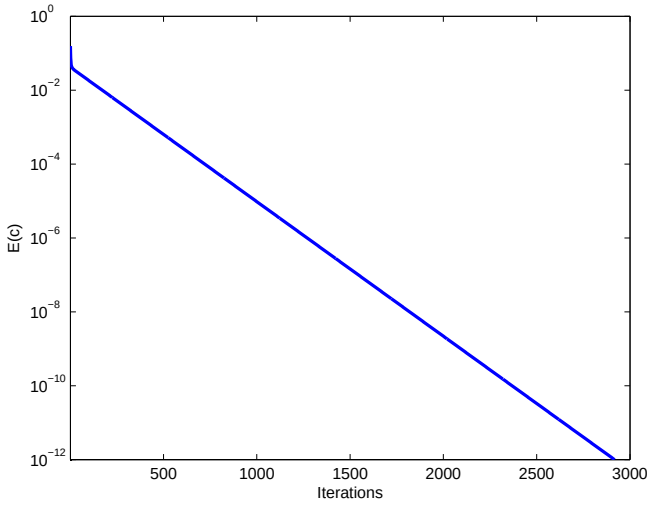


Figure 4.14: Convergence speed of the distributed subgradient projected method via $E(c)$

Therefore, the set of distinct Laplacian eigenvalues is $\mathbf{S}_2 = \{1, 0.3333, 0.25\}$ which deduces $\Lambda = \{1, 1, 3, 3, 4\}$ through Algorithm 10.

3. ADMM-based method

Figure 4.15 shows the trajectory of the coefficients c_k obtained by means of Algorithms 6.

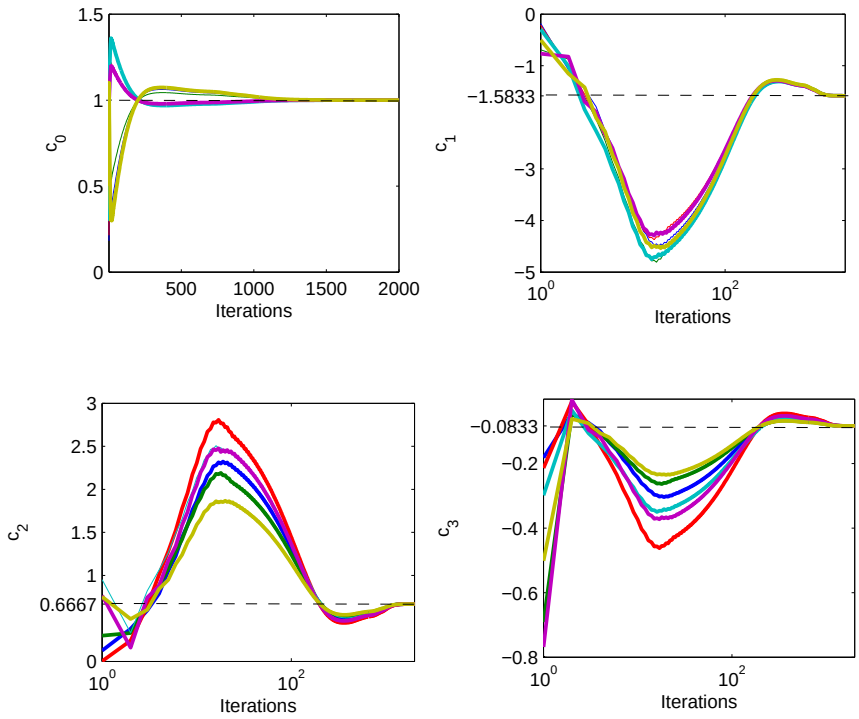


Figure 4.15: Trajectory of the estimated coefficients c_k at each node obtained by means of distributed ADMM

Due to the fact that $D = 3$ is assumed to be known *in a priori*, then the achieved coefficients $\mathbf{c} = [1, -1.5833, 0.6667, -0.0833]^T$ is a unique solution. The convergence is performed in Figure 4.16 with respect to two values of $\rho = \{0.1, 0.01\}$.

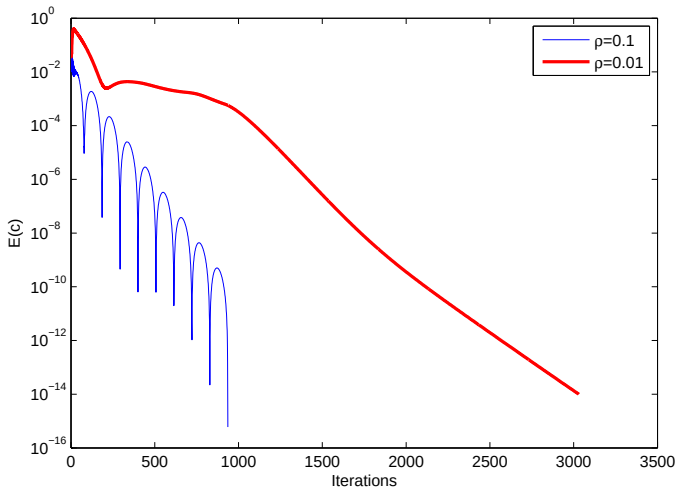


Figure 4.16: Convergence of the distributed ADMM-based method via $E(c)$

Since all three methods infer the same spectrum of Laplacian matrix $sp(\mathbf{L})$, we can estimate the robustness indices by using equations (1.3.1) and (1.3.2):

- Number of spanning trees $\hat{\xi}$:

$$\hat{\xi} = \frac{1}{N}\prod_{i=2}^{N}\lambda_i(\mathbf{L}) = 6 \tag{4.4.3}$$

- Effective graph resistance $\hat{\mathcal{R}}$:

$$\hat{\mathcal{R}} = \sum_{1\leq i,j\leq N}\mathcal{R}_{ij} = N\sum_{i=2}^{N}\frac{1}{\lambda_i(\mathbf{L})} = 17.5 \tag{4.4.4}$$

To conclude, we can see that, network (a), which has a larger number of edges, is more robust than the network (b).

4.4.2 Imperfect consensus value $\bar{x}$

The performance is evaluated by means of the mean square error (MSE) between the estimated Laplacian eigenvalues $\hat{\lambda}_{j,i}$ and the actual ones.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}(\lambda_j - \hat{\lambda}_{j,i})^2. \tag{4.4.5}$$

4.4.2.1 Example 1

Firstly, we run an average consensus protocol using the constant edge weights consensus protocol 1.2.9 with $\alpha = 0.2$ as a stepsize, and $\mathbf{x}(0) = [0.4401, 0.5271, 0.4574, 0.8754, 0.5181, 0.9436]^T$ as an initial condition.

Figure 4.17 depicts the trajectory of the state of each node.

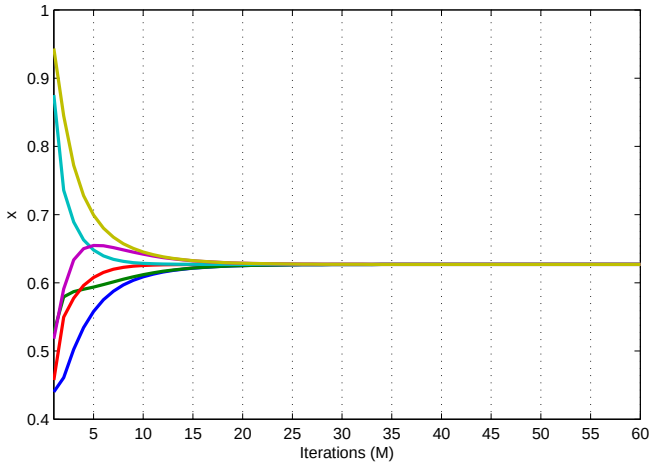


Figure 4.17: Trajectory of the network state during average consensus protocol.

We can note that a reasonable agreement is obtained after iteration 22. Then, Algorithm 7 for computing the factorization of the averaging matrix was performed for different values for the number of iterations M of the average consensus algorithm.

For $M \leq 22$, $\mathbf{x}_M$ was too far from $\bar{\mathbf{x}}$. As a consequence, the obtained polynomial gave rise to negative or complex-valued roots meaning that the coefficients $\tilde{\mathbf{c}}$ created from these roots do not satisfy $\hat{\tilde{x}}_i = \tilde{\mathbf{c}}_i^T \mathbf{q}_i$ in Algorithm 9. For $M > 22$, Algorithm 9 was successfully applied for retrieving the Laplacian eigenvalues approximately.

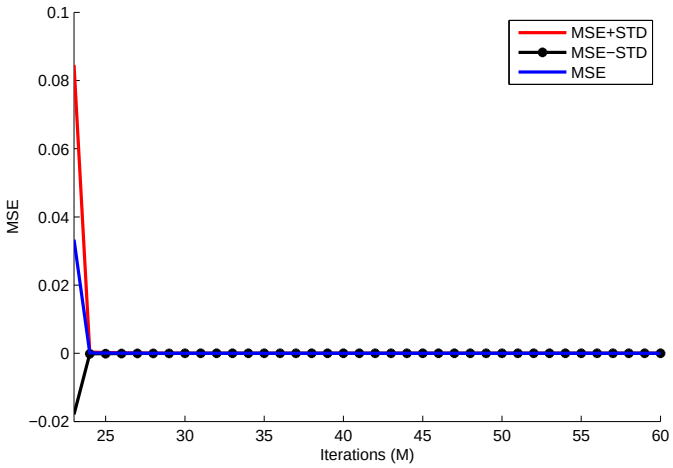


Figure 4.18: Performance of Laplacian eigenvalues estimation for different values of the number of iterations of the standard average consensus algorithm.

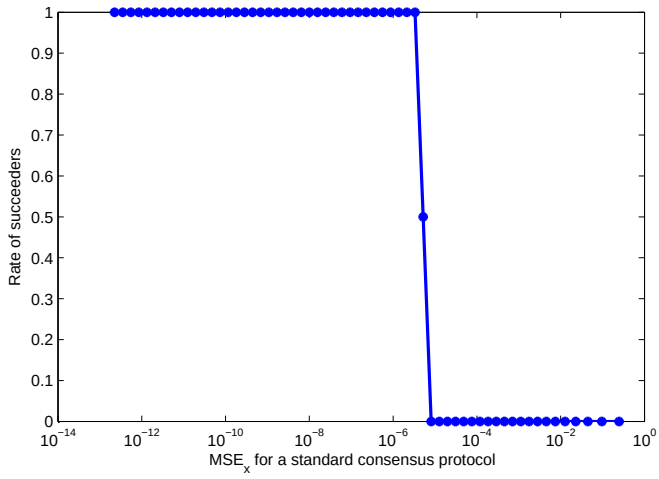


Figure 4.19: Rate of nodes succeeded to compute the Laplacian eigenvalues.

Figure 4.18 depicts the MSE on the estimation of the Laplacian eigenvalues and the corresponding standard deviation. As expected, we can note that the closer $\mathbf{x}_M$ is to $\bar{\mathbf{x}} = 0.6269$, the higher is the precision on the computation of the Laplacian eigenvalues. In addition, the tight bounds around the MSE inform us about the agreement on the estimated Laplacian eigenvalues.

We can note that when the consensus value is close to the actual average, the Laplacian eigenvalues are very well estimated and the agreement on the estimated eigenvalues is perfect. Figure 4.19 shows that when the consensus value is far from the actual average

the nodes fail to compute the Laplacian eigenvalues. However, some of them are able to carry out such a computation while the other fail. For instance, at iteration $M = 22$, there are only 3 nodes estimating the inverse of the Laplacian eigenvalues approximately.

Figures 4.20 and 4.21 show the trajectories of each node on the network robustness indices $(\mathcal{R}, \xi)$ corresponding to the iterations of the standard consensus protocol (M), started from $M = 23$.

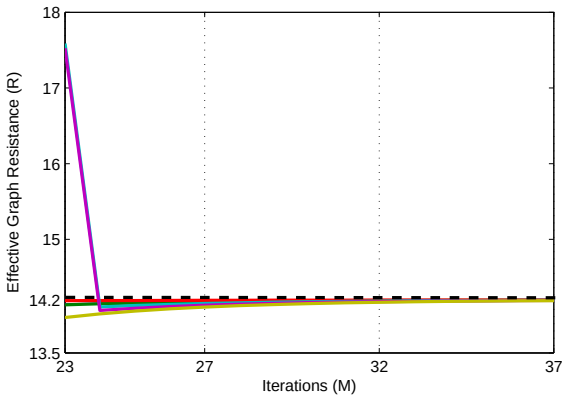


Figure 4.20: Trajectory of each node on the effective graph resistance $\mathcal{R}$

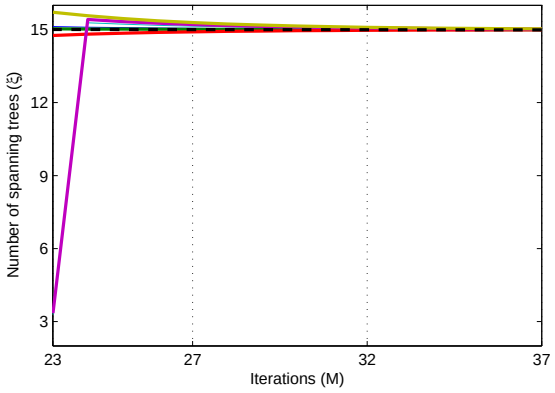


Figure 4.21: Trajectory of each node on the number of spanning trees ξ

From these two Figures, the nodes estimate different values of $\mathcal{R}$ and ξ until $M = 35$. Then, they estimate almost the same value of the Robustness Indices.

Figure 4.22 describes the sensitiveness of the effective graph resistance $(\mathcal{R})$ and the number spanning trees (ξ) to the MSE via the relative errors which are defined respectively as:

$$\begin{aligned} RE_{\mathcal{R}} &= \frac{\|\mathbf{1}\mathcal{R} - \hat{\boldsymbol{\mathcal{R}}}\|^2}{\mathcal{R}^2}, \\ RE_{\xi} &= \frac{\|\mathbf{1}\xi - \hat{\boldsymbol{\xi}}\|^2}{\xi^2}. \end{aligned}$$

Where $\hat{\boldsymbol{\mathcal{R}}}, \hat{\boldsymbol{\xi}}$ are the estimated vectors of local estimated Effective Graph Resistance $\mathcal{R}_i, i = 1, \ldots, N$ and locally estimated the number of spanning trees $\xi_i, i = 1, \ldots, N$ respectively.

Figure 4.22 shows that the effective graph resistance $\mathcal{R}$ is more sensitive to the MSE than the number of spanning trees ξ.

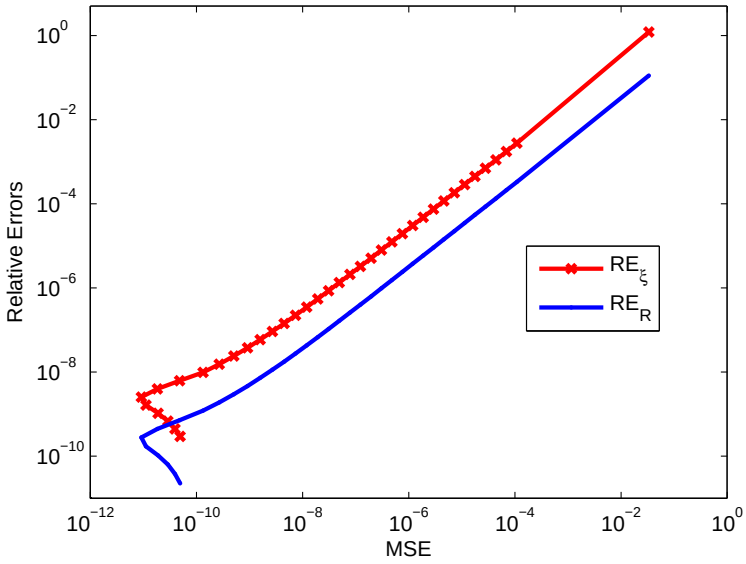


Figure 4.22: Relative errors $RE_{\mathcal{R}}$ and RE_{ξ} versus MSE.

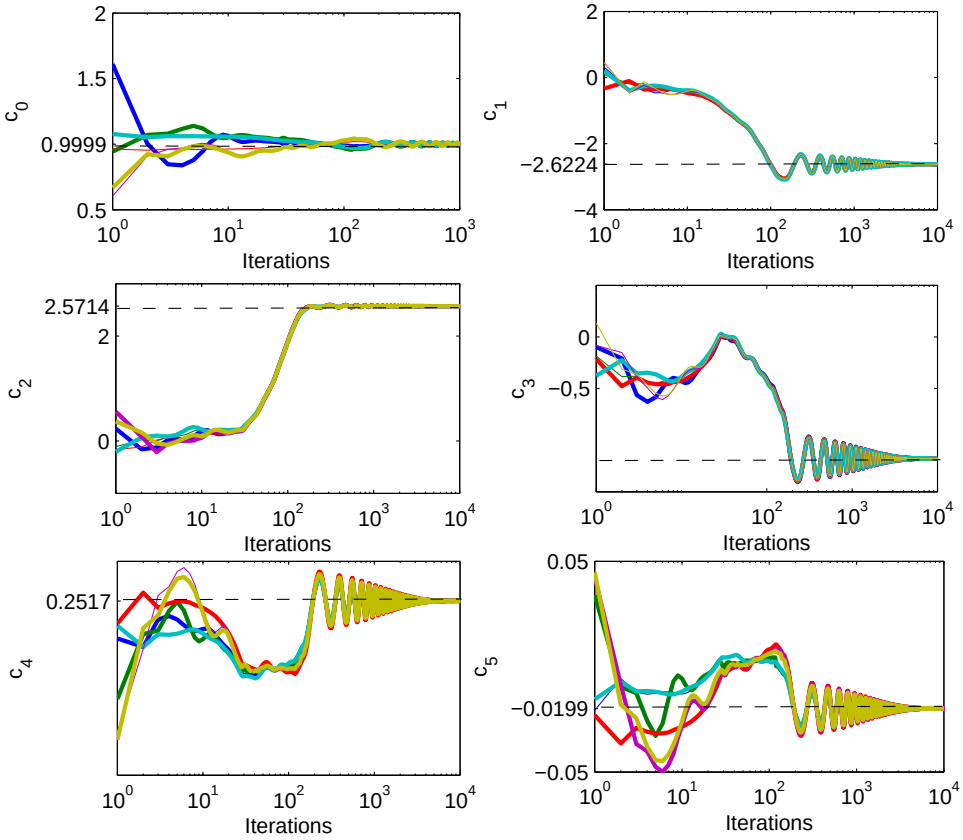


Figure 4.23: Estimation of the polynomial coefficients c_k at $M = 24$.

Let see the estimation of the polynomial coefficients c_k at $M = 24$ via Figure 4.23. We can see that, for some imperfect value of $\bar{x}$ close enough to the actual consensus value $\bar{x}$, we still assess approximately the network robustness.

4.4.2.2 Example 2

For this network, the diameter $d(G) = 3$ can be known *in a priori*. Therefore, we can set $h = d(G) = 3$.

As depicted in Figure 4.24, we first run an average consensus protocol using the constant edge weights consensus protocol 1.2.9 with $\alpha = 0.2$ as a stepsize and set $\mathbf{x}(0) = [0.2785, 0.5469, 0.9575, 0.9649, 0.1576, 0.9706]^T$ as an initial condition. Figure 4.24 depicts the trajectory of the state of each node.

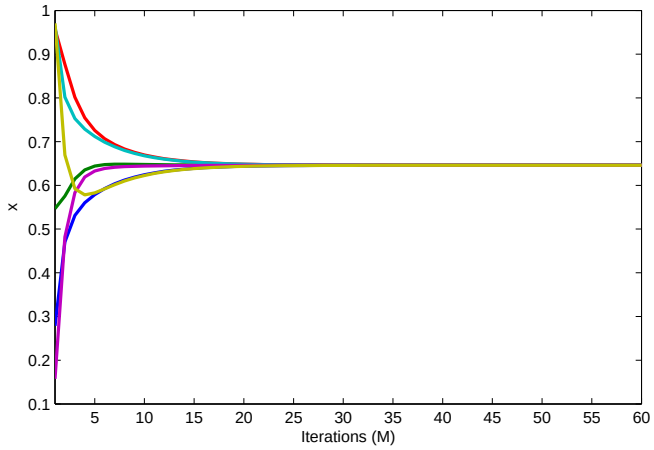


Figure 4.24: Trajectory of the network state during average consensus protocol.

We can note that a reasonable agreement is obtained from iteration 7.

For instance, Figure 4.25 depicts the trajectories of the estimated polynomial coefficients. We can note the good agreement between nodes on the estimation of these polynomial coefficients, which are then used to build a 3rd-order polynomial whose roots contain the inverse of the Laplacian eigenvalues.

For $M > 7$, Algorithm 9 was successfully performed for retrieving the Laplacian eigenvalues. Figure 4.26 depicts the MSE on the estimation of the Laplacian eigenvalues and the corresponding standard deviation. As expected, we can note that the closer $\mathbf{x}_M$ is to $\bar{\mathbf{x}} = 0.646$, the higher is the precision on the computation of the Laplacian eigenvalues. In addition, the tight bounds around the MSE inform us about the agreement on the estimated Laplacian eigenvalues. We can note that when the consensus value is close to

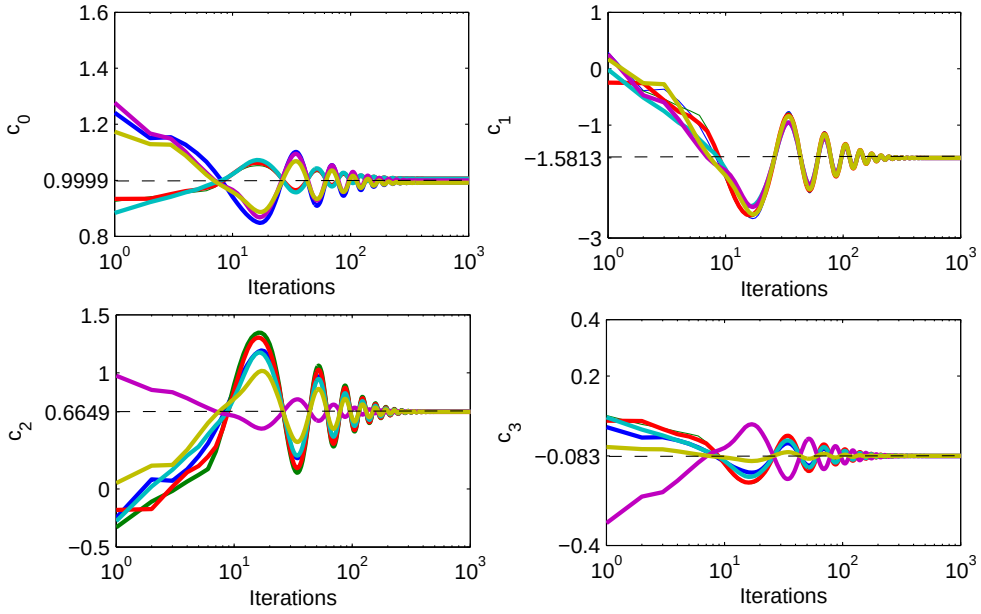


Figure 4.25: Estimation of the polynomial coefficients for ($M = 17$).

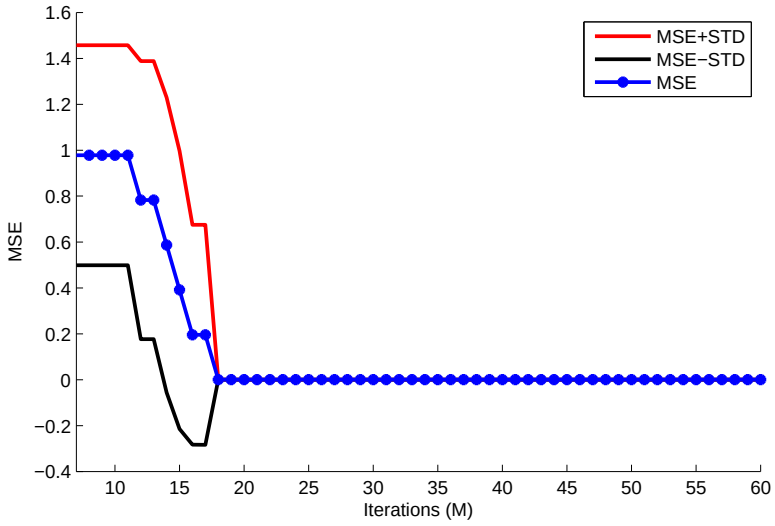


Figure 4.26: Performance of Laplacian eigenvalues estimation for different values of the number of iterations of the standard average consensus algorithm.

the actual average, the Laplacian eigenvalues are very well estimated and the agreement on the estimated eigenvalues is perfect. Figure 4.27 shows that when the consensus value is far from the actual average ($M < 7$) the nodes fail to compute the Laplacian eigenvalues. However, some of them are able to carry out such a computation while the other fail ($M = 7, \ldots, 17$).

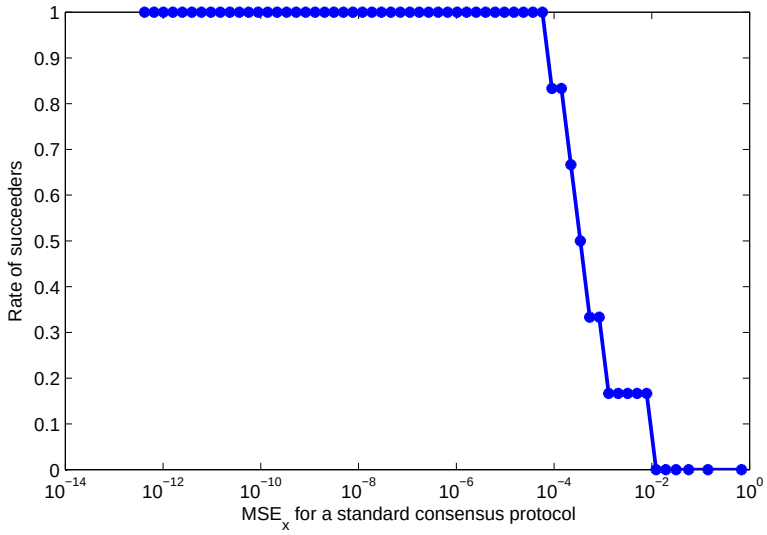


Figure 4.27: Rate of the nodes succeed to compute the Laplacian eigenvalues.

Since started from $M = 18$, all nodes can compute the robustness metrics, then Figures 4.28 and 4.29 demonstrate the trajectories of each node versus iterations ($M = \{18, \ldots, 60\}$).

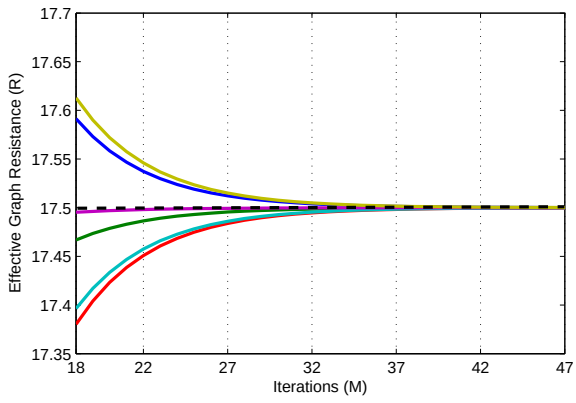


Figure 4.28: Trajectory of each node on the effective graph resistance $\mathcal{R}$

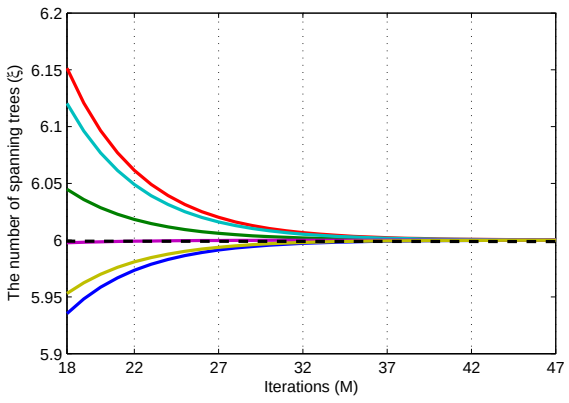


Figure 4.29: Trajectory of each node on the number of spanning trees ξ

Furthermore, Figure 4.30 describes the comparison between two robustness metrics on the sensitiveness to the MSE. Figure 4.30 shows that the effective graph resistance $\mathcal{R}$ is more sensitive to the MSE than the number of spanning trees ξ.

In this example, since the number of the distinct Laplacian eigenvalues is known, then the solution of the polynomial coefficients c_k is unique. Moreover, the convergence of Algorithm 7 assessed through the performance of the cost function $E(c)$ due to different values of M is presented in Figure 4.31. We can see that when the choice of M make x_M

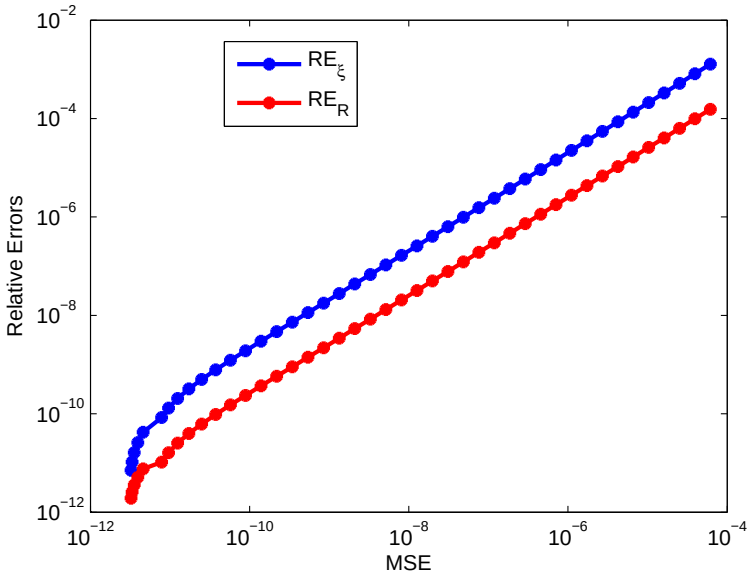


Figure 4.30: Relative Errors $RE_{\mathcal{R}}$ and RE_{ξ} versus MSE.

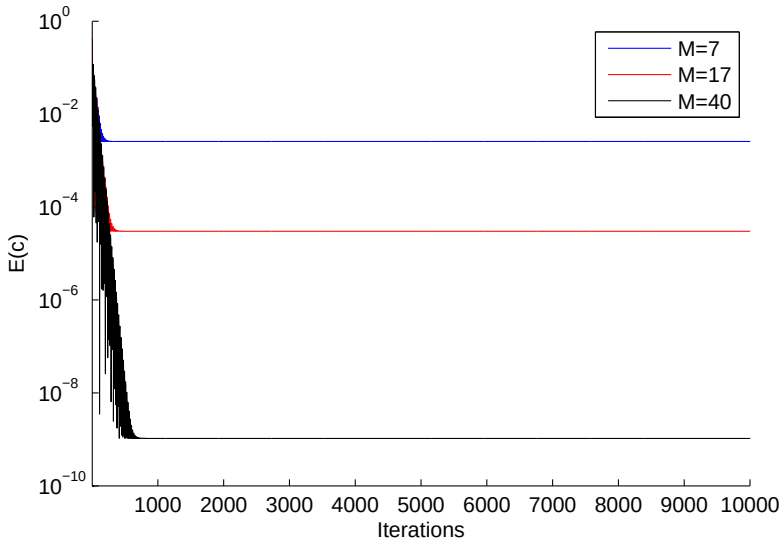


Figure 4.31: Convergence of Algorithm 7 for distinct Laplacian estimation for $M = \{7, 17, 40\}$.

be far from the actual value $\bar{x}$, the convergence of Algorithm 7 is not enough to deduce the exact polynomial of c_k that can be used to infer the Laplacian spectrum.

4.4.3 Unknown consensus value $\bar{x}$

4.4.3.1 Example 1

We first generate the vector of initial values $\mathbf{x}(0) = [0.5356, 0.1649, 0.8834, 0.6665, 0.8477, 0.7627]^T$.

According to the proposed Algorithm 8, the nodes have to achieve agreement on the values of the polynomial coefficients c_k and that of the average consensus value $\bar{x}$ simultaneously.

Figure 4.32 and Figure 4.33 depict the trajectories of the nodes for the estimation of the coefficients $c_{i,k}, k = 0, \ldots, N-1$, and that of the consensus value $\bar{x}$. Note that, the average consensus value is obtained much more faster than the common values of the coefficients c_k.

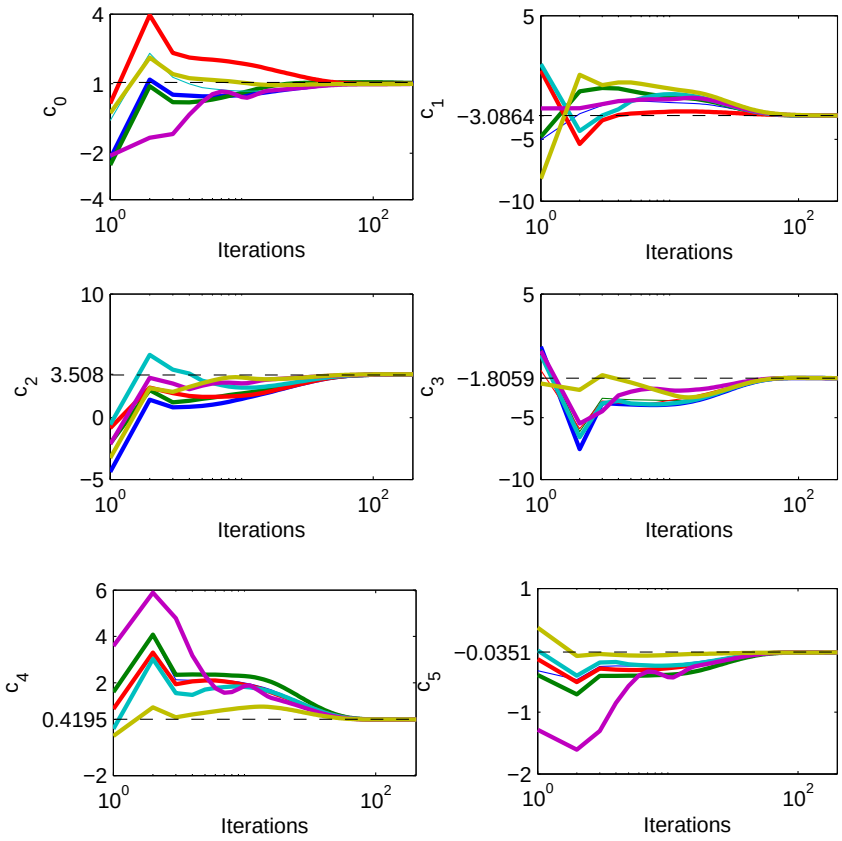


Figure 4.32: Nodes trajectories for the estimation of the coefficients c_k.

To assess the independence of the estimates on the initial condition $\mathbf{x}(0)$, a Monte-Carlo simulation has been carried out. 100 initial vectors $\mathbf{x}(0)$ drawn independently from a uniform distribution have been generated.

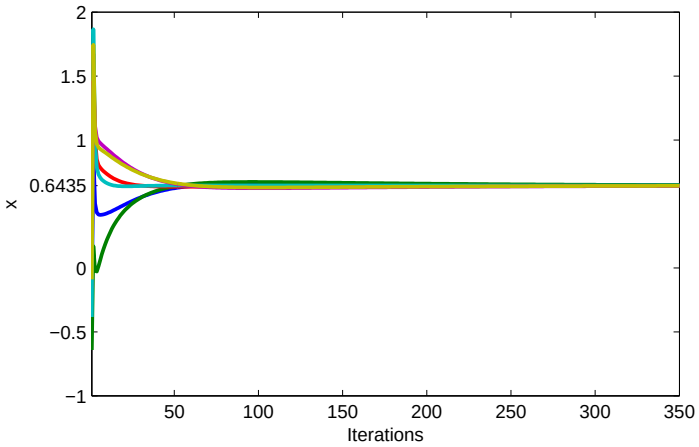


Figure 4.33: Nodes trajectories converging to the average of the initial condition.

Figure 4.34 describes the performance in terms of the MSE between the estimated values and the actual values for the polynomial coefficients $\mathbf{c}_i$ and the average consensus values $\bar{x}_i$ as:

$$MSE_c = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathbf{c}_i - \tilde{\mathbf{c}}\|^2}{\|\tilde{\mathbf{c}}\|^2}, \quad \text{and} \quad MSE_{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N}\frac{(\bar{x}_i - \tilde{x})^2}{\tilde{x}^2},$$

where $\tilde{x}, \tilde{\mathbf{c}}$ stand for the actual average consensus value and actual polynomial coefficients respectively.

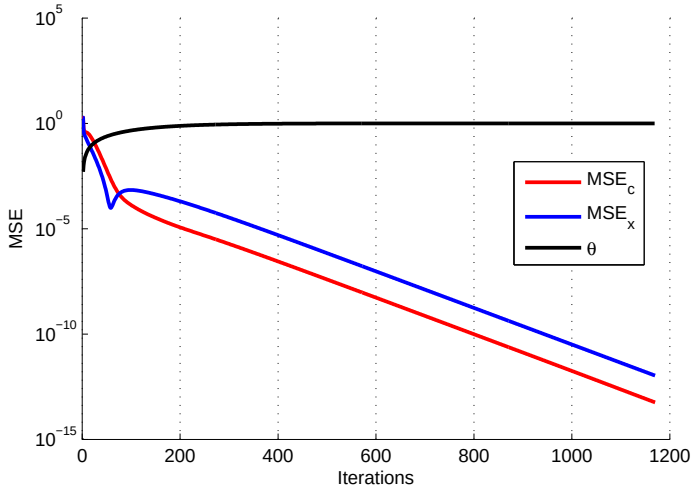


Figure 4.34: Mean square error between the estimated values and actual values with respect to $\mathbf{c}_i$ (MSE_c) and $\bar{x}_i$ (MSE_x).

Figure 4.34 indicates the performances of MSE_c, MSE_x with respect to the variation of $\theta(t)$. As can be seen from the figure, when $\theta(t)$ is near to 0, the convergence speed of

the average consensus problem is faster. However, when $\theta(t)$ tends to 1, the convergence speed of MSE_c becomes faster. On the other hand, the choice of $\theta(t)$ also influences on the convergence speed of the Algorithm 8.

Table 4.1 describes the the achievement of the (4.3.46) with respect to different values of β. As can be seen from this Table, all of them is shown to obtain the nonzero distinct Laplacian eigenvalues.

Table 4.1: The achievement of the proposed algorithms with respect to the variation of β

β	0.1	0.01	0.06
Estimated Laplacian eigenvalues	$\{1, 2, 3, 5\}$		
Estimated coefficient $\mathbf{c}_k$	0.9995	1	0.9998
	-2.8378	-3.0864	-2.7880
	3.0039	3.5080	2.9015
	-1.4674	-1.8059	-1.3984
	0.3287	0.4195	0.3102
	-0.0269	-0.0351	-0.0252
Number of iterations	301	1170	259
$\mathcal{R}$	14.2		
ξ	15		

Moreover, with the same choice of penalty parameter $\rho = 0.1$ and threshold (stopping criterion), regarding to the accuracy of the estimation of the coefficients c_t, the choice of value $\beta = 0.01$ is the best one. Besides that, all of the solutions of the polynomial coefficients c_k give the same assessment of the network robustness. Regarding to the speed of the convergence, $\beta = 0.06$ seems to be the good choice in this example.

4.4.3.2 Example 2

We first generate the initial values $\mathbf{x}(0) = [0.6531, 0.0403, 0.5047, 0.8945, 0.3857, 0.2921]^T$.

According to the proposed Algorithm 8, the nodes have to achieve agreement on the values of the polynomial coefficients c_k and that of the average consensus value $\bar{x}$.

Figure 4.35 and Figure 4.36 depict the trajectories of the nodes for the estimation of the coefficients $c_{i,k}, k = 0, \ldots, N-1$, and that of the consensus value $\bar{x}$. All the nodes agree on the actual values of polynomial coefficients $c_{i,k}$ and average consensus value $\bar{x}$, and convergence occurs in less than 100 iterations. Moreover, the average consensus value is obtained much more faster than the common values of the coefficients c_k.

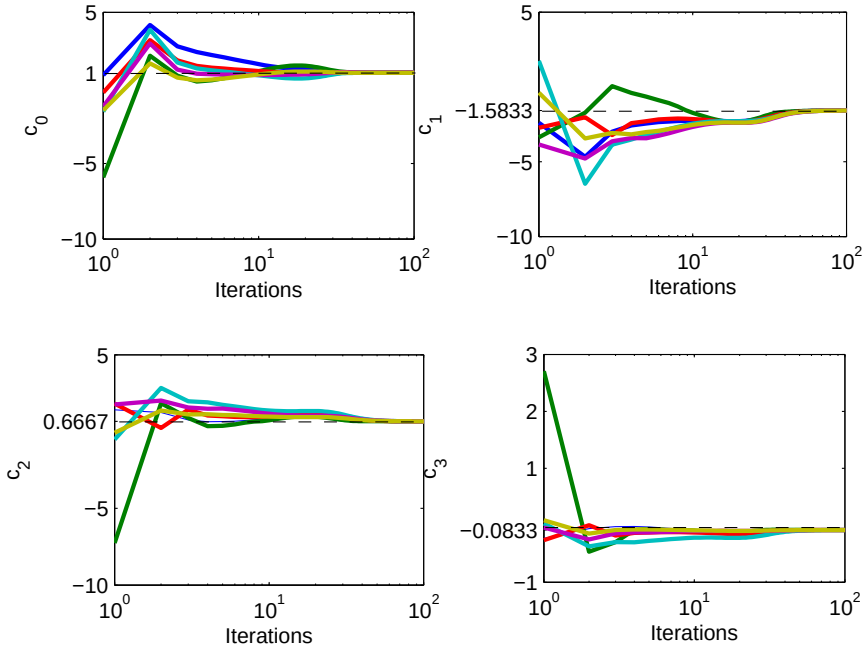


Figure 4.35: Nodes trajectories for the estimation of the coefficients c_k.

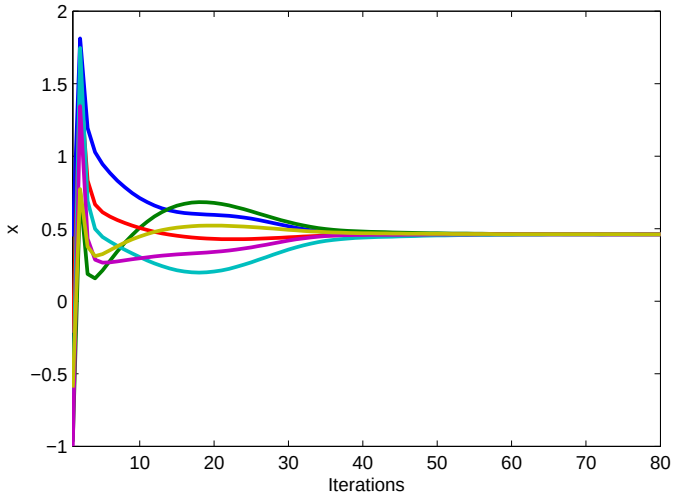


Figure 4.36: Nodes trajectories converging to the average of the initial condition.

Table 4.2 shows the performance of the proposed method (Algorithm 8) with respect to different choice of the β.

Table 4.2: The achievement of the proposed algorithms with respect to the variation of β

β	0.1	0.01	0.06
Estimated Laplacian eigenvalues	$\{1, 3, 4\}$		
Estimated coefficient $\mathbf{c}_k$	1 -1.5833 0.6667 -0.0833		
Number of iterations	151	1188	232
$\mathcal{R}$	17.5		
ξ	6		

As can be seen from this Table, despite of the different choices of β, the proposed method gives exactly the same polynomial coefficients c_k, which can be used to infer the same assessment of the network robustness. However, with $\beta = 0.1$, the speed of convergence is the fastest.

4.5 Conclusion

In this Chapter, we have proposed some methods to assess the robustness of a given network in a distributed way. Formally, the robustness metrics called the effective graph resistance $\mathcal{R}$ and the number of spanning trees ξ are functions of the Laplacian spectrum $sp(\mathbf{L})$. Therefore, these proposed methods are distinguished by the way to estimate the Laplacian spectrum. In detail, three cases have been considered with respect to the consensus value $\bar{x}$, that are: perfect consensus value $\bar{x}$, imperfect consensus value $\bar{x}_M$ and unknown consensus value $\bar{x}$. Based on recent results, the average consensus matrix can be factored in D Laplacian based consensus matrices, where D stands for the number of nonzero distinct Laplacian eigenvalues. For this purpose, all proposed methods are characterized into two approaches, which are the direct approach and indirect approach in the case of perfect consensus value $\bar{x}$. A direct approach was first presented. It gives rise to a non-convex optimization problem with respect to the stepsizes $\alpha_k, k = 1, \ldots, D = N-1$ that are the inverse of the nonzero distinct Laplacian eigenvalues. For this approach, since it is based on the gradient descent method, the main issue is that it suffers from slow convergence towards a local minimum. To improve that, a convex optimization problem has been considered to estimate the optimal coefficients $c_k, k = 0, \ldots, N-1$ of a given polynomial whose roots are the stepsizes $\alpha_k, k = 1, \ldots, N-1$ that allow to factorize the averaging matrix. In this way, we have applied two well-known methods that are the projected subgradient method and the ADMM to solve the convex problem. After that, the Laplacian eigenvalues are thus estimated by taking the inverse of these

obtained stepsizes $\alpha_k, k = 1, \ldots, N-1$. Then the real set of nonzero distinct Laplacian eigenvalues $\Lambda = \{\lambda_i, i = 2, \ldots, D\}$ is captured by a retrieving algorithm. Finally, the whole Laplacian spectrum is obtained via defining the corresponding multiplicities of these eigenvalues. In what follows, the network robustness measure can be easily deduced.

The drawback of the proposed methods is their scalability. In fact, for large graphs, the whole Laplacian spectrum estimation becomes a significant burden. Therefore, the estimation can be restricted to that of the most significant eigenvalues used to estimate bound of the network robustness index. For example, $\lambda_2(\mathbf{L})$ is a lower-bound of the vertex connectivity ($\mathcal{K}$). Furthermore, we may encounter ill-conditioned problem in implementation of the indirect proposed methods. Hence, we can investigate new method for preconditioning the problem.

In the case of imperfect consensus value, we assume that a standard consensus protocol has been run and stores some intermediate values of $\bar{x}_M$ according to the iteration M. Then, we make use of the ADMM-based Algorithm 7 to see how the network robustness reacts. Finally, it claims that when the intermediate value of $\bar{x}$ is as close as possible to the real one, the network robustness index can be taken approximately.

In the case of unknown consensus value $\bar{x}$, we consider the convex optimization problem which resorts to a convex combination of two convex functions. The first one takes into account the average consensus problem while the second one concerns the estimation of the Laplacian matrix. It is an indirect method where parameters of the factorization of the averaging matrix are computed first and then the Laplacian eigenvalues are estimated as the roots of a given polynomial. For large graphs, such a solution is clearly intractable. Therefore, the proposed Algorithms 5, 6, 7, 8 are efficient and well adapted for networks with medium size.

The ADMM seems to be the best method compared with all the methods proposed. Therefore, the study of the convergence rate must be more interesting. In the near future, we will investigate the convergence rate of the ADMM for the quadratic programming based on the work in [25].

Chapter 5

Network topology reconstruction

Contents

5.1 Introduction

Network topology is usually a schematic description of the arrangement of a network, including its nodes and connecting edges. Briefly speaking, network topology may describe how the data is transferred between these nodes.[1]

Network topology identification refers to detecting and identifying the interested network elements and the relationship between elements in target network and represents the topology construction in an appropriate form. Therefore, identification of networks of systems becomes a growing attractive task for solving many problems in different science and engineering domains, for instance, in biology (biochemical, neural and ecological

[1]Source: http://whatis.techtarget.com/definition/network-topology

networks [2]), finance, computer science (Internet and World Wide Web), transportation (delivery and distribution network), and electrical engineering.

For example, the architecture of an overlay network - how it allocates addresses, etc.- may be significantly optimized by knowledge of the distribution and connectivity of the nodes on the underlay network that actually carries the traffic. Several important systems, such as P4P (stands for provider portal for Peer–to–Peer (P2P) applications) and RMTP (reliable multicast transport protocol), utilize information about the topology of the underlay network for optimization as well as management [1].

In the Internet, for instance, the usual mechanism for generating the topology of a network is by the use of Traceroute. Traceroute is executed on a node, called the source, by specifying the address of a destination node. This execution produces a sequence of identifiers, called a trace, corresponding to the route taken by packets travelling from the source to the destination. A trace set T is generated by repeatedly executing Traceroute over a network, varying the terminal nodes, i.e. the source and destination. If T contains traces that identify every instance when an edge is incident on a node, it is possible to reconstruct the network exactly [1].

Following our study, from a general consensus network and consensus measurements, we deal with the problem of inferring the correct network topology in the presence of anonymous nodes (or unlabelled nodes). In other words, is it possible for an arbitrary node j to reconstruct the correct network from average consensus measurements?

5.2 Literature review

In the literature, there are various methods proposed for reconstructing the network topology.

For centralized framework, in [56], the authors introduced a network identification scheme which involves the excitation and observation of nodes running consensus-type protocols. The problem is to identify the interaction geometry among a known number of nodes, adopting a (weighted) consensus-type algorithm for their coordination. In the considered procedure, the node broadcasts a zero state to its neighbors without being removed from the network. This node is called node knockout, which is essentially a grounding procedure. Starting with the number of vertices in the network as a known parameter as well as the controllability and observability of the resulting steered-and-observed network, define the characteristic polynomial of the grounded consensus at assumed known node i, which leads to deduce the matrix $\mathbf{\Psi}$ as the adjugate of the matrix $(\mathbf{I}_N + \mathbf{L})$ in order to determine the entries of $\mathbf{L}$. The advantage of this paper is the ramifications for exact identification from boundary nodes of networks that have an embedded consensus-based distributed algorithm used for flocking and distributed estimation.

In [68], the problem of identifying the topology of an unknown directed network of linear time-invariant (LTI) systems stimulated by a wide-sense stationary noise of unknown power spectral density has been considered. The authors have proposed several reconstruction algorithms based on the power spectral properties of the network response to the noise. In [33], the identifiability of the structure and dynamics of an unknown network driven by unknown noise has been assessed based on factorizations of the output spectral density. In this paper, the authors have claimed that for networks with closed-loop transfer matrices that are minimum phase, the network reconstruction problem can have a unique solution from its output spectral density.

In [37], the reconstruction concepts are based on paradigmatic dynamical models such as phase oscillators. The authors have discovered a general connection between dynamical correlation among oscillators and the underlying topology in the presence of noise. In particular, the correspondence between the dynamical correlation matrix of nodal time series and the connection matrix of structures, due to the presence of noise.

In order to bridge dynamical correlation and topology, consider N non-identical coupled oscillators $\mathbf{x}_i \in I\!R^m, i = 1, \ldots, N$, which denotes the m-dimensional state variables of i-th oscillator. In the presence of noise, the dynamics of the whole coupled-oscillator system can be expressed as:

$$\dot{\mathbf{x}}_i = \mathbf{F}_i(\mathbf{x}_i) - \gamma \sum_{j=1}^{N} l_{ij}\mathbf{P}(\mathbf{x}_j) + \boldsymbol{\eta}_i, \tag{5.2.1}$$

where:

- $\mathbf{F}_i : I\!R^m \to I\!R^m$ is intrinsic dynamics of the i-th oscillator,
- $\mathbf{P} : I\!R^m \to I\!R^m$ is the coupling function of oscillators,
- l_{ij} is an element at i-th row and j-th column of the Laplacian matrix $\mathbf{L}$,
- $\gamma > 0$ is the coupling strength,
- $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_N]^T$ is the noise vector with covariance matrix $\hat{\mathbf{D}}$.

Let $\bar{\mathbf{x}}_i$ be the counterpart of $\mathbf{x}_i$ in the absence of noise and assume a small perturbation ξ_i, then $\mathbf{x}_i = \bar{\mathbf{x}}_i + \xi_i$. Substituting this into (5.2.1), the variational equation can be obtained:

$$\dot{\boldsymbol{\xi}} = (D\mathbf{F}(\bar{\mathbf{x}}) - \gamma \mathbf{L} \otimes D\mathbf{P}(\bar{\mathbf{x}}))\,\dot{\boldsymbol{\xi}} + \boldsymbol{\eta},$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^T$ denotes the deviation vector, $D\mathbf{F}(\bar{\mathbf{x}}) = diag\,(D\mathbf{F}_1(\bar{\mathbf{x}}_1), \ldots, D\mathbf{F}_N(\bar{\mathbf{x}}_N))$ and $D\mathbf{F}_i, D\mathbf{P}$ are $m \times m$ Jacobian matrices of $\mathbf{F}_i$

and $\mathbf{P}$ respectively. Denoting $\mathbf{C} \triangleq \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \boldsymbol{\xi}\boldsymbol{\xi}^T dt$ the dynamical correlation of oscillators, with $[0, \mathcal{T}]$ is the time interval in which the time average is performed, we have:

$$\mathbf{0} = \frac{d}{dt}\left[\frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} \boldsymbol{\xi}\boldsymbol{\xi}^T dt\right] = -\mathbf{MC} - \mathbf{CM}^T + \frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} \boldsymbol{\eta}\boldsymbol{\xi}^T dt + \frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} \boldsymbol{\xi}\boldsymbol{\eta}^T dt, \tag{5.2.2}$$

where $\mathbf{M} = -\left(D\mathbf{F}(\bar{\mathbf{x}}) + \gamma\mathbf{L} \otimes D\mathbf{P}(\bar{\mathbf{x}})\right)$. After a few basic manipulations, (5.2.2) can be rewritten as:

$$\mathbf{MC} + \mathbf{CM}^T = \hat{\mathbf{D}}. \tag{5.2.3}$$

This equation illustrates a general relationship between the dynamical correlation $\mathbf{C}$ and the Laplacian $\mathbf{L}$ in the presence of noise as characterized by $\hat{\mathbf{D}}$. If we consider one-dimensional state variable and linear coupling such that $D\mathbf{P} = 1$, with Gaussian white noise $\hat{\mathbf{D}} = \sigma^2\mathbf{I}_{mN}$, and further regard the intrinsic dynamics $D\mathbf{F}$ as small perturbations. Then, (5.2.3) can be simplified to $\mathbf{LC} + \mathbf{CL}^T = \sigma^2\frac{\mathbf{I}_{mN}}{\gamma}$. For an undirected network with symmetric coupling matrix, the solution of $\mathbf{C}$ can be expressed as

$$\mathbf{C} = \frac{\sigma^2}{2\gamma}\mathbf{L}^+, \tag{5.2.4}$$

where $\mathbf{L}^+$ denotes the pseudo inverse of the Laplacian matrix.

Hence, the dynamic correlation matrix $\mathbf{C}$ is closely related to the network connection matrix $\mathbf{L}$, which can be used to infer network structures when no knowledge about nodal dynamics is available.

From (5.2.4), the Laplacian $\mathbf{L}$ can be extracted from the knowledge of noise-corrupted time series:

$$\mathbf{L} = \frac{\sigma^2}{2\gamma}\mathbf{C}^+, \tag{5.2.5}$$

where $\mathbf{C}^+$ is the pseudo inverse. The correlation matrix is inversely proportional to the Laplacian matrix that contains full information about the network structure. The reconstruction of the network topology based on time series then becomes possible for undirected networks. In case of directed networks, there is no unique solution for $\mathbf{L}$ from $\mathbf{C}$, because the asymmetric $\mathbf{L}$ has twofold degree of freedom as that of symmetric of $\mathbf{C}$. Thus, the global structure of directed networks cannot be inferred solely depending on the correlation, but we can still infer the local structure.

Based from this method, in [55] the authors have presented a fully distributed algorithm for reconstructing the topology of an undirected consensus network by using a distributed least-squares algorithm.

Assume the number of nodes N is known, the noisy consensus dynamics is given by:

$$\dot{\mathbf{x}} = -\gamma\mathbf{L}\mathbf{x} + \boldsymbol{\eta}, \tag{5.2.6}$$

(5.2.7)

where $\mathbf{x} = [x_1, \ldots, x_N]^T \in I\!R^N$ and $\boldsymbol{\eta} \in I\!R^N$.

From (5.2.5), we obtain:

$$\mathbf{L} = \frac{\sigma^2}{2\gamma}\mathbf{C}^+ = \left[\frac{\sigma^2}{2\gamma}\mathbf{C} + \mathbf{J}_N\right]^{-1} - \mathbf{J}_N, \tag{5.2.8}$$

where $\mathbf{J}_N$ is the averaging matrix. And $\mathbf{C}$ is obtained as:

$$\mathbf{C} = \frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} (\mathbf{x}(t) - \bar{x}\mathbf{1})(\mathbf{x} - \bar{x}\mathbf{1})^T dt$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Therefore,

$$\mathbf{L} = \left[\frac{\sigma^2}{2\gamma}(\mathbf{I}_N - \mathbf{J}_N)\mathbf{C}_R(\mathbf{I}_N - \mathbf{J}_N) + \mathbf{J}_N\right]^{-1} - \mathbf{J}_N, \tag{5.2.9}$$

where

$$\mathbf{C}_R = \frac{1}{\mathcal{T}}\int_0^{\mathcal{T}} \mathbf{x}(t)\mathbf{x}^T(t)dt.$$

Assuming $\mathbf{s}$ discrete observations in the time interval $[0, \mathcal{T}]$, $\mathbf{C}_R$ can be expressed approximately as:

$$\mathbf{C}_R \simeq \frac{1}{\mathbf{s}}\sum_{k=0}^{\mathbf{s}-1} \mathbf{x}(k)\mathbf{x}^T(k).$$

In order to estimate the $\mathbf{C}_R$, the measurement equation for node i is introduced:

$$\mathbf{y}_i(k) = \mathbf{E}_i\mathbf{x}(k)$$

where $\mathbf{E}_i \in \{0,1\}^{(|N_i|+1)\times N}$ is a selection matrix. By collecting the measurements of all nodes at time k, $\mathbf{y}(k) = \mathbf{E}\mathbf{x}(k), k \in \{0, 1, \ldots, S-1\}$ where $\mathbf{y} = [\mathbf{y}_1^T \ \ldots \ \mathbf{y}_N^T]^T$ and $\mathbf{E} = [\mathbf{E}_1^T \ \ldots \ \mathbf{E}_N^T]^T$.

If $\mathbf{E}^T\mathbf{E}$ is invertible, then the least-squares estimate of $\mathbf{x}(k)$ is

$$\hat{\mathbf{x}}(k) = \mathbf{E}^+\mathbf{y}(k) = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T\mathbf{y}(k) = \left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{E}_i^T\mathbf{E}_i\right]^{-1}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{E}_i^T\mathbf{y}_i(k)\right]$$

Let assume node i maintains a matrix $\mathbf{B}_i \in I\!R^{N\times N}$ and a vector $\mathbf{z}_i \in I\!R^N$ and executes the following iterations for $h \in \{0, 1, \ldots, D-1\}, D > 1$

$$\mathbf{B}_i(h+1) \ = \ \mathbf{B}_i(h) + \zeta\sum_{j\in N_i}(\mathbf{B}_j(h) - \mathbf{B}_i(h)), \tag{5.2.10}$$

$$\mathbf{z}_i(h+1) \;=\; \mathbf{z}_i(h) + \zeta \sum_{j \in N_i} (\mathbf{z}_j(h) - \mathbf{z}_i(h)), \tag{5.2.11}$$

where $\mathbf{B}_i(0) = \mathbf{E}_i^T \mathbf{E}_i$ and $\mathbf{z}_i(0) = \mathbf{E}_i^T \mathbf{y}_i(k), i = 1, \ldots, N$. If $\zeta \in (0, 1)$ satisfies the condition $\zeta < \frac{2}{\lambda_N(\mathbf{L})}$, then

$$\lim_{h \to \infty} \mathbf{B}_i(h) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{E}_i^T \mathbf{E}_i \tag{5.2.12}$$

$$\lim_{h \to \infty} \mathbf{z}_i(h) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{E}_i^T \mathbf{y}_i(k), \tag{5.2.13}$$

and node i asymptotically obtains an estimate $\hat{\mathbf{x}}_i$ of vector $\mathbf{x}$ at time k:

$$\hat{\mathbf{x}}_i(k) = \lim_{h \to \infty} \mathbf{B}_i^{-1}(h) \mathbf{z}_i(h).$$

Node i can then compute its local estimate $\mathbf{C}_{R,i}$ of $\mathbf{C}_R$ as:

$$\hat{\mathbf{C}}_{R,i} = \frac{1}{\mathbf{s}} \sum_{k=0}^{\mathbf{s}-1} \hat{\mathbf{x}}_i(k) \hat{\mathbf{x}}_i^T(k).$$

and eventually obtains the Laplacian matrix through (5.2.9).

In [40], the authors have proposed a decentralized reconstruction algorithm using an algebraic method using observability properties of a network. Consider a connected undirected graph $G(V, E)$ with the dynamics presented as:

$$\begin{aligned} \mathbf{X}(k+1) &= \mathbf{W}\mathbf{X}(k). \\ \mathbf{Y}_i(k) &= \mathbf{E}_i \mathbf{X}(k). \end{aligned}$$

Where:

- $\mathbf{X}(k) \in I\!R^{N \times M}$ is a matrix with $\mathbf{x}_i(k), i = 1, \ldots, N$ as rows;
- $\mathbf{Y}_i(k)$ is the outputs or node values that are seen by node i at the k^{th} time step;
- $\mathbf{E}_i \in I\!R^{\bar{d}_i \times N}$ is the row selection matrix with $\bar{d}_i$ in the interval $[0, d_i + 1]$;
- $\mathbf{W} = \mathbf{I}_N - \epsilon \mathbf{L}$ is the consensus matrix. Its eigenvalue decomposition is $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where the eigenvector and eigenvalues are respectively organized in the orthogonal matrix $\mathbf{U}$ and the diagonal matrix $\mathbf{D}$.

The identification of the network topology is implemented by estimating both eigenvalues (in Chapter 4.2) and eigenvectors of the network matrix. From the observability theory,

the pair $(\mathbf{W}, \mathbf{E}_i)$ is observable if and only if the observability matrix $\mathbf{O}_{i,K_i} \in I\!R^{(K_i+1)\bar{d}_i \times N}$ defined as:

$$\mathbf{O}_{i,K_i} = \begin{pmatrix} \mathbf{E}_i \\ \mathbf{E}_i\mathbf{W} \\ \mathbf{E}_i\mathbf{W}^2 \\ \vdots \\ \mathbf{E}_i\mathbf{W}^{K_i} \end{pmatrix} = \begin{pmatrix} \mathbf{V}_i \\ \mathbf{V}_i\mathbf{D} \\ \vdots \\ \mathbf{V}_i\mathbf{D}^{K_i} \end{pmatrix} \mathbf{U}^T = \bar{\mathbf{O}}_{i,K_i}\mathbf{U}^T$$

with $\mathbf{V}_i = \mathbf{E}_i\mathbf{U}$ is full column rank, i.e $(\mathbf{O}_i) = N$.

From the available observations $\mathbf{Y}_i(k), k = 0, 1, \ldots, K_i + 1$, and the two generated matrices:

$$\bar{\mathbf{Y}}_i = \begin{pmatrix} \mathbf{Y}_i(0) \\ \mathbf{Y}_i(1) \\ \vdots \\ \mathbf{Y}_i(K_i) \end{pmatrix}; \quad \bar{\bar{\mathbf{Y}}}_i = \begin{pmatrix} \mathbf{Y}_i(1) \\ \mathbf{Y}_i(2) \\ \vdots \\ \mathbf{Y}_i(K_i+1) \end{pmatrix},$$

we can show that

$$\begin{aligned} \bar{\mathbf{Y}}_i &= \bar{\mathbf{O}}_{i,K_i}\mathbf{C}^T \\ \bar{\bar{\mathbf{Y}}}_i &= \bar{\mathbf{O}}_{i,K_i}\mathbf{D}\mathbf{C}^T, \end{aligned}$$

with $\mathbf{C}^T = \mathbf{U}^T\mathbf{X}(0)$.

As presented in Chapter 4.2, $\tilde{\mathbf{U}}_1, \tilde{\mathbf{U}}_2$ are respectively the top and the bottom parts of $\tilde{\mathbf{U}}$ of left singular vectors of matrix $(\bar{\mathbf{Y}}_i^T\bar{\bar{\mathbf{Y}}}_i^T)^T$ corresponding to $\tilde{\mathbf{U}}_1 = \bar{\mathbf{O}}_{i,K_i}\mathbf{T}^T$ and $\tilde{\mathbf{U}}_2 = \bar{\mathbf{O}}_{i,K_i}\mathbf{D}\mathbf{T}^T$.

By defining $\mathbf{G} = \mathbf{T}\bar{\mathbf{O}}_{i,K_i}^T\bar{\mathbf{O}}_{i,K_i}$, we get $\mathbf{R}_1 = \mathbf{G}\mathbf{T}^T$ and $\mathbf{R}_2 = \mathbf{G}\mathbf{D}\mathbf{T}^T$. Then, with $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1^{-1}$, we get $\mathbf{R}\mathbf{G} = \mathbf{G}\mathbf{D}$, which is a standard eigenvalue problem. Then, $\mathbf{D}, \mathbf{G}$ can be deduced.

Then, we can estimate $\mathbf{T}^T = \mathbf{G}^{-1}\mathbf{R}_1$ and $\bar{\mathbf{O}}_{i,K_i} = \mathbf{R}_1^{-1}\mathbf{G}\tilde{\mathbf{U}}_1$. As a result, $\mathbf{C}^T = \mathbf{G}^{-1}\mathbf{R}_1\bar{\mathbf{Y}}_1$.

In order to infer the structure of the network, assume that each node knows $\mathbf{E}_i, \mathbf{C}, \mathbf{D}$, the authors try to compute $\mathbf{U}$.

At first, build the matrices $\mathbf{\Phi}_i$ and the vectors $\mathbf{\Psi}_i$ defined as:

$$\mathbf{\Phi}_i = \begin{pmatrix} \mathbf{C} \otimes \mathbf{E}_i \\ \mathbf{C}\mathbf{D} \otimes \mathbf{E}_i \\ \vdots \\ \mathbf{C}\mathbf{D}^{K_i} \otimes \mathbf{E}_i \end{pmatrix}, \quad \mathbf{\Psi}_i = \begin{pmatrix} vec(\mathbf{Y}_i(0)) \\ vec(\mathbf{Y}_i(1)) \\ \vdots \\ vec(\mathbf{Y}_i(K_i)) \end{pmatrix}$$

Then, compute the local quantities $\mathbf{\Phi}_i^T\mathbf{\Phi}_i$ and $\mathbf{\Phi}_i^T\mathbf{\Psi}_i$ with an average consensus algorithm to get:

$$\begin{aligned}\mathbf{\Phi} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{\Phi}_i^T\mathbf{\Phi}_i \\ \mathbf{\Psi} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{\Phi}_i^T\mathbf{\Psi}_i.\end{aligned}$$

Finally, by solving the least-squares problem $\mathbf{\Phi}vec(\mathbf{U}) = \mathbf{\Psi}$, the matrix $\mathbf{U}$ can be computed.

At the end, each node can estimate the consensus matrix as $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, and this is the network topology.

In all these methods, the presence of anonymous nodes is not considered.

5.3 Problem statement

From the previous chapter we now know that node i can obtain the overall Laplacian spectrum from average consensus measurements. We state the following assumption:

Assumption 1: All the nonzero Laplacian eigenvalues are distinct.

By considering a network represented by a graph $G(V, E)$, we recall that at time instant k, the state of this network is given by the vector

$$\mathbf{x}(k) = \mathbf{W}\mathbf{x}(k-1)$$

or equivalently

$$\mathbf{x}(k) = \mathbf{W}^k\mathbf{x}(0)$$

Let us consider the following eigenvalue decomposition of the consensus matrix $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T, \tag{5.3.1}$$

with $\mathbf{D} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$ and $\mathbf{\Delta}$ is the diagonal matrix of Laplacian eigenvalues. ϵ can be chosen as $\frac{2}{\lambda_2(\mathbf{L})+\lambda_N(\mathbf{L}}$ [74], then $\lambda_i(\mathbf{W}) = 1 - \epsilon\lambda_i(\mathbf{L}), i = 1, \ldots, N$.

Since the Laplacian eigenvectors form a basis of $\mathbb{R}^N$, the initial condition $\mathbf{x}(0)$ can be expanded such that $\mathbf{x}(0) = \sum_{i=1}^{N}\beta_i\mathbf{U}_{.i} = \mathbf{U}\mathbf{b}$, where $\mathbf{b} = \begin{pmatrix}\beta_1 & \beta_2 & \cdots & \beta_N\end{pmatrix}^T$ contains the expansion coefficients, and $\mathbf{U}_{.i}$ stands for the i-th eigenvector, i.e. the i-th column of $\mathbf{U}$. Therefore:

$$\mathbf{x}(k) = \mathbf{U}\mathbf{D}^k\mathbf{U}^T\mathbf{U}\mathbf{b} = \mathbf{U}\mathbf{D}^k\mathbf{b} = \mathbf{U}diag(\mathbf{b})vecd(\mathbf{D}^k),$$

where $vecd(\cdot)$ is the column vector built with the diagonal entries of the matrix in the argument.

We can equivalently write the above equation as:

$$\mathbf{x}(k) = \tilde{\mathbf{U}} vecd(\mathbf{D}^k), \quad \text{with} \quad \tilde{\mathbf{U}} = \mathbf{U} diag(\mathbf{b})$$

Therefore, $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T = \mathbf{U} diag(\mathbf{b}.^2)\mathbf{U}^T$ and $\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} = diag(\mathbf{b}.^2)$. We can conclude that:

$$\mathbf{W} = \tilde{\mathbf{U}}\mathbf{D} diag(\mathbf{b}.^2)^{-1}\tilde{\mathbf{U}}^T. \tag{5.3.2}$$

5.4 Distributed solution to the network topology reconstruction problem

5.4.1 Estimation of the eigenvectors of the Laplacian-based consensus matrix

At node j, the state at instant k is then given by:

$$x_j(k) = \sum_{i=1}^{N} \lambda_i^k(\mathbf{W})\beta_i u_{j,i},$$

$u_{j,i}$ being the jth entry of the eigenvector $\mathbf{U}_{.i}$. By stacking N consecutive measurements node j obtains:

$$\begin{pmatrix} x_j(0) \\ x_j(1) \\ \vdots \\ x_j(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \lambda_2(\mathbf{W}) & \cdots & \lambda_N(\mathbf{W}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_2^{N-1}(\mathbf{W}) & \cdots & \lambda_N^{N-1}(\mathbf{W}) \end{pmatrix} \begin{pmatrix} \tilde{u}_{j,1} \\ \tilde{u}_{j,2} \\ \vdots \\ \tilde{u}_{j,N} \end{pmatrix}, \quad \tilde{u}_{j,i} = u_{j,i}\beta_i$$

or equivalently

$$\mathbf{x}_j = \boldsymbol{\Upsilon}\tilde{\mathbf{U}}_{j.}^T \tag{5.4.1}$$

From Assumption 1, we can conclude that $\boldsymbol{\Upsilon}$ is a full rank matrix. Therefore node j can efficiently solve the above problem and get:

$$\tilde{\mathbf{U}}_{j.}^T = \boldsymbol{\Upsilon}^{-1}\mathbf{x}_j. \tag{5.4.2}$$

Now, the node only knows the jth row of $\tilde{\mathbf{U}}$.

Let us consider a message passing scheme where the nodes send to their neighbors the rows they have. Then after a number of message exchanges, at least equal to the diameter of the graph, node j has all the rows of $\tilde{\mathbf{U}}$. However, since some of the nodes are assumed to be anonymous, they do not send the label of the row they have. Two kinds of messages are transmitted in the network. The rows associated to non-anonymous nodes are correctly labelled while those associated to anonymous nodes are not labelled. As a consequence, node j only receives $\tilde{\mathbf{U}}$ up to rows permutation, i.e. it receives $\hat{\mathbf{U}} = \mathbf{\Pi}\tilde{\mathbf{U}}$, $\mathbf{\Pi}$ being a permutation matrix.

5.4.2 Network topology reconstruction

We can note that $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = diag(\mathbf{b}^2)$. Therefore, from (5.3.2), we get:

$$\mathbf{W} = \mathbf{\Pi}\hat{\mathbf{W}}\mathbf{\Pi}^T, \quad \text{with} \quad \hat{\mathbf{W}} = \hat{\mathbf{U}}\mathbf{D}(\hat{\mathbf{U}}^T\hat{\mathbf{U}})^{-1}\hat{\mathbf{U}}^T. \tag{5.4.3}$$

We can conclude that from the estimated matrix $\hat{\mathbf{W}}$, we obtain a graph that is isomorphic to the original graph. In what follows we state two sufficient conditions to ensure correct reconstruction of the graph. We first recall the following notions of observability:

A network is said to be [44]:

- node-observable from a given node if that node is able to reconstruct the entire network state from its own measurements. This issue has been studied for instance in [58] and [65] where it has been stated that a network with a consensus matrix having at least one non-simple eigenvalue is not node-observable.
- neighborhood observable from a given node if that node can reconstruct the entire network state from its own measurements and those of its neighbors. This issue was studied in [43].
- globally observable if it is neighborhood observable from any node.

Proposition 4

Assume that the graph is node-observable or neighborhood observable from node j. If all the entries of the initial condition are distinct then the network topology can be exactly reconstructed from node j.

Proof: From $\tilde{\mathbf{U}}$ and $diag(\mathbf{b}^2)$ the initial condition $\mathbf{x}(0)$ can be reconstructed as follows: $\mathbf{x}(0) = \tilde{\mathbf{U}}diag(\mathbf{b}^2)\boldsymbol{\beta}$, with $\boldsymbol{\beta} = \begin{pmatrix} 1/\beta_1^2 & 1/\beta_2^2 & \cdots & 1/\beta_N^2 \end{pmatrix}^T$. Since $\tilde{\mathbf{U}}$ is known only up to rows permutation through $\hat{\mathbf{U}}$, the reconstructed initial condition $\hat{\mathbf{x}}(0) = \hat{\mathbf{U}}diag(\mathbf{b}^2)\boldsymbol{\beta}$ is related to the actual initial condition as follows: $\hat{\mathbf{x}}(0) = \mathbf{\Pi}\mathbf{x}(0)$. The graph being node-observable or neighborhood-observable from node j, node j can reconstruct the

initial condition from its consecutive measurements. It can therefore compare $\hat{\mathbf{x}}(0)$ with $\mathbf{x}(0)$. If all the entries of $\mathbf{x}(0)$ are distinct then the permutation matrix can be retrieved and the matrix $\mathbf{W}$ exactly recovered from $\hat{\mathbf{W}}$. ■

Proposition 5

Assume that the network contains a set $\mathcal{A}$ of anonymous nodes and the graph is node-observable or neighborhood observable from node j. If all the entries of the initial condition associated with the anonymous nodes are all distinct then the network topology can be exactly reconstructed from node j.

The proof of this proposition is similar to the previous one. Here, the permutation ambiguity is restricted to the entries associated with the anonymous nodes.

We can also note that if the initial condition is driven from a continuous joint probability distribution, then the reconstruction of the network topology is almost sure.

The proposed network topology reconstruction algorithm is described as followed:

Algorithm 11 (*Network topology reconstruction*)

1. *Inputs:*
   - *Given the Laplacian spectrum $sp(\boldsymbol{L})$ and the number of nodes N.*
   - *Given a stepsize ϵ of consensus protocol with matrix $\boldsymbol{W} = \boldsymbol{I}_N - \epsilon \boldsymbol{L}$.*
   - *Given the first $N-1$ consensus measurements $\mathbf{x}_i = [x_i(0), x_i(1), \ldots, x_i(N-1)]^T, i = 1, \ldots, N$.*
2. *Initialization:*
   - *Set of eigenvalues $\lambda_i(\boldsymbol{W}) = 1 - \epsilon\lambda_i(\boldsymbol{L})$ for $i = 1, \ldots, N$.*
   - *Matrix $\boldsymbol{\Upsilon}$ in equation (5.4.1).*
3. *Each node j estimates the j-th row of $\tilde{\boldsymbol{U}}$ using (5.4.2).*
4. *Each node sends to its neighbors the rows it has. After a number of message exchanges, each node j obtains $\hat{\boldsymbol{U}} = \boldsymbol{\Pi}\tilde{\mathbf{U}}$.*
5. *Since $diag(\boldsymbol{b}^2) = \hat{\boldsymbol{U}}^T \hat{\boldsymbol{U}}$ is a diagonal matrix, then each node can then estimate the matrix topology as $\boldsymbol{W} = \boldsymbol{\Pi}\left(\hat{\boldsymbol{U}}\boldsymbol{D}diag(\hat{\boldsymbol{U}}^T\hat{\boldsymbol{U}})^{-1}\hat{\boldsymbol{U}}^T\right)\boldsymbol{\Pi}^T$*
6. *Define the estimated Laplacian matrix $\hat{\boldsymbol{L}} = \frac{1}{\epsilon}(\boldsymbol{I}_N - \boldsymbol{W})$. Then take the rounded estimated Laplacian matrix to see the effectiveness of the proposed method.*

5.5 Numerical result

In this section, we first consider the case of a network as depicted in Figure 5.1.

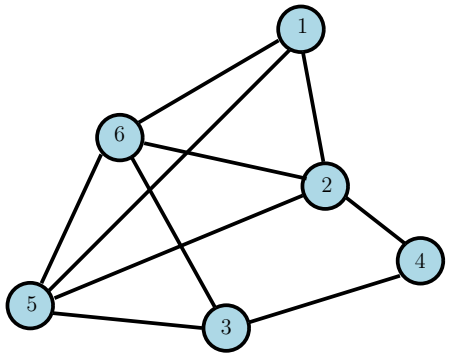


Figure 5.1: An generic network with 6 nodes.

The Laplacian matrix is defined as:

$$\mathbf{L} = \begin{pmatrix} 3 & -1 & 0 & 0 & -1 & -1 \\ -1 & 4 & 0 & -1 & -1 & -1 \\ 0 & 0 & 3 & -1 & -1 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & -1 \\ -1 & -1 & -1 & 0 & -1 & 4 \end{pmatrix}$$

The corresponding Laplacian spectrum of this graph is $sp(\mathbf{L}) = \{0, 1.7857, 3, 4.5392, 5, 5.6751\}$.

We consider two cases: perfect Laplacian eigenvalues $sp(\mathbf{L})$ and two imperfect Laplacian spectrum $\tilde{sp}(\mathbf{L})$ to see how Algorithm 11 reacts.

At first, we run the Algorithms 7, 9 and 10 to obtain the perfect Laplacian spectrum $sp(\mathbf{L})$ and the two arbitrary imperfect Laplacian spectrums $\tilde{sp}_1(\mathbf{L}), \tilde{sp}_2(\mathbf{L})$. Hence, we can deduce the eigenvalues of consensus matrix $\mathbf{W}$ corresponding to both imperfect cases:

$sp(\mathbf{W})$	$\tilde{sp}_1(\mathbf{W})$	$\tilde{sp}_2(\mathbf{W})$
1	1	1
0.5213	0.5211	0.5185
0.1958	0.1967	0.1847
-0.2168	-0.2161	-0.1945
-0.3403	-0.3403	-0.3403
-0.5213	-0.5225	-0.5151

From now, we can build the matrix $\boldsymbol{\Upsilon}$ and $\tilde{\boldsymbol{\Upsilon}}_1$, $\tilde{\boldsymbol{\Upsilon}}_2$ as follows:

$$\boldsymbol{\Upsilon} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0.5213 & 0.1958 & -0.2168 & -0.3403 & -0.5213 \\ 1 & 0.2718 & 0.0383 & 0.0470 & 0.1158 & 0.2718 \\ 1 & 0.1417 & 0.0075 & -0.0102 & -0.0394 & -0.1417 \\ 1 & 0.0739 & 0.0015 & 0.0022 & 0.0134 & 0.0739 \\ 1 & 0.0385 & 0.0003 & -0.0005 & -0.0046 & -0.0385 \end{pmatrix}$$

$$\tilde{\boldsymbol{\Upsilon}}_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0.5211 & 0.1967 & -0.2161 & -0.3403 & -0.5225 \\ 1 & 0.2715 & 0.0387 & 0.0467 & 0.1158 & 0.2730 \\ 1 & 0.1415 & 0.0076 & -0.0101 & -0.0394 & -0.1427 \\ 1 & 0.0737 & 0.0015 & 0.0022 & 0.0134 & 0.0745 \\ 1 & 0.0384 & 0.0003 & -0.0005 & -0.0046 & -0.0390 \end{pmatrix}$$

$$\tilde{\boldsymbol{\Upsilon}}_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0.5185 & 0.1847 & -0.1945 & -0.3403 & -0.5151 \\ 1 & 0.2688 & 0.0341 & 0.0378 & 0.1158 & 0.2653 \\ 1 & 0.1394 & 0.0063 & -0.0074 & -0.0394 & -0.1367 \\ 1 & 0.0723 & 0.0012 & 0.0014 & 0.0134 & 0.0704 \\ 1 & 0.0375 & 0.0002 & -0.0003 & -0.0046 & -0.0363 \end{pmatrix}$$

Next, we take the first $N-1$ consecutive measurements $\mathbf{x}_i$ that can be expressed in matrix form:

$$\mathbf{X} = \begin{pmatrix} 0.9933 & 0.5656 & 0.6131 & 0.5651 & 0.5574 & 0.5477 \\ 0.3567 & 0.5455 & 0.5618 & 0.5332 & 0.5492 & 0.5378 \\ 0.7529 & 0.4524 & 0.5382 & 0.5148 & 0.5371 & 0.5332 \\ 0.1100 & 0.3485 & 0.4291 & 0.4939 & 0.5101 & 0.5278 \\ 0.5970 & 0.6360 & 0.5588 & 0.5635 & 0.5445 & 0.5466 \\ 0.4306 & 0.6927 & 0.5395 & 0.5700 & 0.5423 & 0.5474 \end{pmatrix}$$

By (5.4.1), each node i gets the row $\tilde{\mathbf{U}}_{i\cdot}$ of the matrix $\tilde{\mathbf{U}}$ and sends it to its neighbors. After a number of message exchanges, we can obtain $\hat{\mathbf{U}}$ and $\hat{\mathbf{U}}_1, \hat{\mathbf{U}}_2$ as:

$$\hat{\mathbf{U}} = \begin{pmatrix} 0.5401 & 0.2146 & -0.0746 & 0.3006 & -0.0000 & 0.0127 \\ 0.5401 & 0.0351 & -0.0746 & -0.2407 & -0.0000 & 0.0969 \\ 0.5401 & -0.1128 & 0.1492 & 0.1109 & -0.0000 & 0.0654 \\ 0.5401 & -0.3624 & -0.0746 & 0.0511 & -0.0000 & -0.0441 \\ 0.5401 & 0.1128 & 0.0373 & -0.1109 & 0.0832 & -0.0654 \\ 0.5401 & 0.1128 & 0.0373 & -0.1109 & -0.0832 & -0.0654 \end{pmatrix}$$

$$\hat{\mathbf{U}}_1 = \begin{pmatrix} 0.5401 & 0.2149 & -0.0750 & 0.2989 & 0.0021 & 0.0124 \\ 0.5401 & 0.0351 & -0.0742 & -0.2417 & 0.0019 & 0.0956 \\ 0.5401 & -0.1131 & 0.1493 & 0.1100 & 0.0020 & 0.0645 \\ 0.5401 & -0.3627 & -0.0740 & 0.0504 & 0.0000 & -0.0437 \\ 0.5401 & 0.1129 & 0.0370 & -0.1088 & 0.0802 & -0.0643 \\ 0.5401 & 0.1129 & 0.0370 & -0.1088 & -0.0862 & -0.0643 \end{pmatrix}$$

$$\hat{\mathbf{U}}_2 = \begin{pmatrix} 0.5402 & 0.2176 & -0.0868 & 0.2757 & 0.0363 & 0.0105 \\ 0.5402 & 0.0340 & -0.0736 & -0.1966 & -0.0564 & 0.1090 \\ 0.5399 & -0.1115 & 0.1513 & 0.0889 & 0.0177 & 0.0666 \\ 0.5399 & -0.3679 & -0.0680 & 0.0368 & 0.0185 & -0.0493 \\ 0.5402 & 0.1140 & 0.0386 & -0.1024 & 0.0751 & -0.0684 \\ 0.5402 & 0.1140 & 0.0386 & -0.1024 & -0.0913 & -0.0684 \end{pmatrix}$$

Finally, each node can estimate the matrix topology $\hat{\mathbf{W}}$ and $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2$ as:

$$\hat{\mathbf{W}} = \begin{pmatrix} 0.1958 & 0.2681 & 0.0000 & -0.0000 & 0.2681 & 0.2681 \\ 0.2681 & -0.0723 & -0.0000 & 0.2681 & 0.2681 & 0.2681 \\ -0.0000 & 0.0000 & 0.1958 & 0.2681 & 0.2681 & 0.2681 \\ -0.0000 & 0.2681 & 0.2681 & 0.4639 & 0.0000 & -0.0000 \\ 0.2681 & 0.2681 & 0.2681 & -0.0000 & -0.0723 & 0.2681 \\ 0.2681 & 0.2681 & 0.2681 & -0.0000 & 0.2681 & -0.0723 \end{pmatrix}$$

$$\hat{\mathbf{W}}_1 = \begin{pmatrix} 0.1958 & 0.2681 & 0.0000 & -0.0000 & 0.2681 & 0.2681 \\ 0.2667 & -0.0724 & 0.0000 & 0.2679 & 0.2721 & 0.2656 \\ -0.0012 & -0.0001 & 0.1958 & 0.2679 & 0.2719 & 0.2657 \\ 0.0026 & 0.2683 & 0.2680 & 0.4641 & -0.0080 & 0.0050 \\ 0.2680 & 0.2681 & 0.2681 & -0.0000 & -0.0722 & 0.2680 \\ 0.2680 & 0.2681 & 0.2681 & -0.0000 & 0.2681 & -0.0723 \end{pmatrix}$$

$$\hat{\mathbf{W}}_2 = \begin{pmatrix} 0.1958 & 0.2681 & 0.0000 & -0.0000 & 0.2681 & 0.2681 \\ 0.2552 & -0.0734 & 0.0005 & 0.2668 & 0.3074 & 0.2437 \\ 0.0274 & 0.0025 & 0.1948 & 0.2707 & 0.1843 & 0.3199 \\ 0.0123 & 0.2692 & 0.2676 & 0.4651 & -0.0376 & 0.0233 \\ 0.2546 & 0.2669 & 0.2685 & -0.0013 & -0.0313 & 0.2427 \\ 0.2546 & 0.2669 & 0.2685 & -0.0013 & 0.3091 & -0.0977 \end{pmatrix}$$

Define the estimated rounded Laplacian matrices $\hat{\mathbf{L}}, \hat{\mathbf{L}}_1, \hat{\mathbf{L}}_2$ as:

$$\hat{\mathbf{L}} = \hat{\mathbf{L}}_1 = \hat{\mathbf{L}}_2 = \begin{pmatrix} 3 & -1 & 0 & 0 & -1 & -1 \\ -1 & 4 & 0 & -1 & -1 & -1 \\ 0 & 0 & 3 & -1 & -1 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & -1 \\ -1 & -1 & -1 & 0 & -1 & 4 \end{pmatrix}$$

We can see that, even with the inexact Laplacian spectrum (with a relevant tolerance of the error between the actual Laplacian spectrum ($\tilde{sp}(\mathbf{L})$)and the real Laplacian spectrum $sp(\mathbf{L})$), the proposed method can infer the network topology.

Now, to achieve a more general evaluation of the proposed algorithm, we generate 100 random graphs with different number of nodes. We assess the quality of network reconstruction for different values of Laplacian eigenvalues estimation defined as

$$Error = \|\tilde{sp}(\mathbf{L}) - sp(\mathbf{L})\|.$$

The performance is evaluated using the Success rate of existing links (SREL) and the Success rate of non-existing links (SRNL), [37]. SREL(SRNL)is defined as the ratio of the number successfully predicted existent (non-existent) links to the total number of existent (nonexistent) links.

Figures 5.2 and 5.3 describe the average SREL and SRNL for graphs of different sizes with respect to the norm of the estimation error of Laplacian eigenvalues.

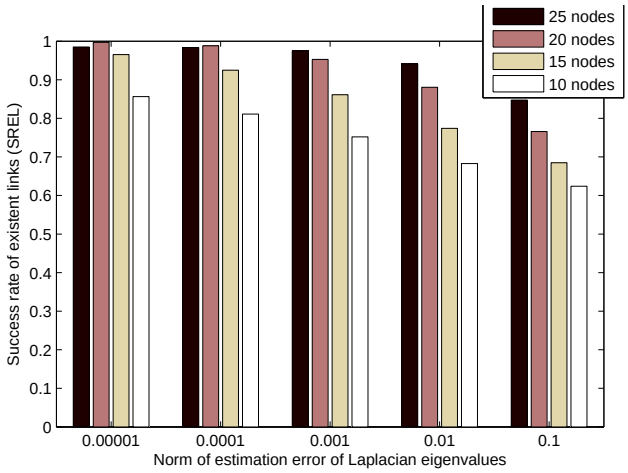


Figure 5.2: Success rate of existing links for 4 graphs with different sizes on the different values of norm of estimation error of Laplacian eigenvalues.

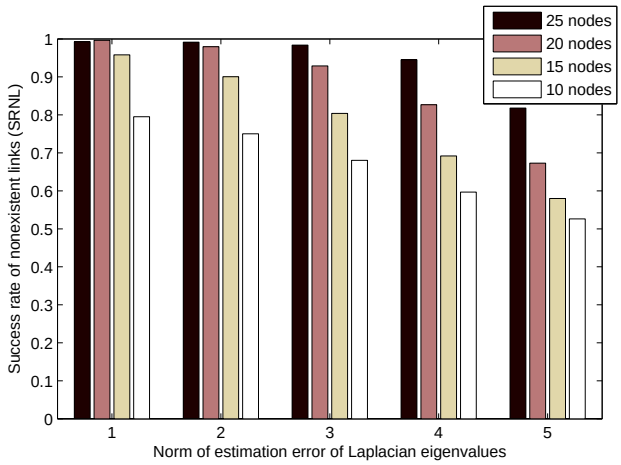


Figure 5.3: Success rate of non-existing links for 4 graphs with different sizes on the different values of norm of estimation error of Laplacian eigenvalues.

As can be seen, whatever the size of the graph, SREL and SRNL decrease with the quality of estimation of Laplacian eigenvalues. In addition, for a given estimation error, the performance also decreases when the size of the graph grows. This is due to the quality of estimation of the eigenvectors. Indeed, by increasing the size of the graph, the rows at the bottom of matrix $\boldsymbol{\Upsilon}$ becomes smaller and smaller and leads to loose the numerical rank. Therefore the quality of estimation can be significantly lost. It is then necessary to resort to more efficient procedure to solve the eigenvector estimation problem for large graphs.

5.6 Conclusion

In this chapter, we have proposed a way for reconstructing the network topology in the presence of anonymous nodes from average consensus measurements. The consensus matrix is determined from its eigenstructure estimated in distributed way. Since the Laplacian spectrum can be obtained by means of the algorithms proposed in the previous chapter, each node can deduce the matrix $\hat{\mathbf{U}}$ up to rows permutation because of anonymous nodes. Therefore, the obtained graph is isomorphic to the original graph. In the case that the graph is node-observable or neighborhood observable from node j, if all the entries of $\mathbf{x}(0)$ are distinct then node j can exactly reconstruct the network topology.

Algorithm 11 works with the assumption that all eigenvalues are distinct. Therefore, future works encompass the design of network reconstruction protocols that deal with spectrum in which the multiplicities of the corresponding eigenvalues can be higher than 1. In addition, numerical issues for large graphs are to be considered for making the proposed method scalable.

Chapter 6

General conclusions and future works

Contents

To conclude this dissertation, we give a brief summary and discussion on future works.

6.1 Review of contributions and conclusions

Finite-time consensus problems have received a vast attention from the research community due to their benefits in many application domains, especially in distributed computation. Briefly speaking, the finite-time average consensus algorithms (for which consensus can be obtained in a finite number of steps) are commonly used as building blocks for more sophisticated distributed control, estimation or inference algorithms. Due to the fact that they can guarantee a minimal execution time, they are much more appealing than those ensuring asymptotic convergence.

This thesis impacts with the design of finite-time average consensus protocols regardless of the Laplacian matrix of a given graph in a distributed way and its application to network robustness assessment.

First, we proposed a new distributed algorithm for self-configuration of finite-time average consensus where matrices are not necessarily based on the Laplacian matrix of a given graph. More precisely, we solved a matrix factorization problem in a distributed way by using a learning sequence. The method is based on the back-propagation method and an associated gradient descent method. Conceptually, each linear iteration $k, k = 0, \ldots, D$

is illustrated as a layer of a network. The main idea is to compute the partial derivatives of the error between the actual output and desired output, and then propagate them back to each iteration k in order to update the weights.

Obviously, this method is impractical for large-scale networks. However, one interesting point can be noted in this work is that the obtained consensus matrices are not doubly stochastic matrices and allow to reach consensus in a minimum number of steps.

Second, the study on the robustness of networks has increasingly attracted the attention of the research community. Therefore, in this thesis, the assessment of the network robustness via the effective graph resistance ($\mathcal{R}$) and the number of spanning trees (ξ) in a distributed way, obviously becomes a promising application domain. Since both robustness metrics are functions of the Laplacian spectrum $sp(\mathbf{L})$, thus the main task is now to estimate the Laplacian spectrum. More precisely, three cases have been considered: perfectly known $\bar{x}$, approximated, and unknown $\bar{x}$.

The average consensus matrix can be factored in D Laplacian based consensus matrices, where D stands for the number of nonzero distinct Laplacian eigenvalues. For the first case (perfectly known $\bar{x}$), all methods are characterized into two approaches, which are direct approach and indirect approach. The direct method gives rise to a non-convex optimization problem with respect to the stepsizes $\alpha_k, k = 1, \ldots, N-1$ that are the inverse of the nonzero distinct Laplacian eigenvalues. Since, this is a gradient based method, it suffers the slow convergence. To improve that, a convex optimization problem is introduced to estimate the coefficients $c_k, k = 0, \ldots, N-1$, of a given polynomial whose roots are the stepsizes $\alpha_k, k = 1, \ldots, N-1$, that allows to factorize the averaging matrix $\mathbf{J}_N = \frac{\mathbf{1}\mathbf{1}^T}{N}$. In this way, two famous methods (the subgradient projected method and ADMM) are applied to solve this convex optimization problem. By means of these proposed methods, the distinct Laplacian eigenvalues can be deduced. Then, by means of an integer programming, the whole spectrum $sp(\mathbf{L})$ is obtained to compute the robustness indices.

The drawback of these methods is scalability. For large-scale graphs, the whole spectrum $sp(\mathbf{L})$ estimation is significantly difficult to obtain. Therefore, the estimation can be restricted to that of the most significant eigenvalues to estimate some bounds of the network robustness indices.

Projected subgradient method is very applicable due to its simplicity. However, the drawback is that it can be slow. Therefore, the choice of the stepsize $\gamma(t)$ is typical critical. In contrast, under mild conditions, the ADMM is guaranteed to converge for all tunning parameters. It has a single penalty parameter ρ that influences on the speed of convergence. Therefore, in the left cases, ADMM is to be the best choice.

In the case of imperfect consensus value $\bar{x}$, we assumed that a standard consensus protocol has been run and stored some intermediate values of $\bar{x}_M$ according to the iteration M.

Then, we made use of the ADMM-based Algorithm 7 to evaluate the network robustness. Finally, it claims that when the intermediate value of $\bar{x}$ is as close as possible to the real one, the network robustness index can be estimated with a very good accuracy.

In the case of unknown consensus value $\bar{x}$, we considered a convex optimization problem which resorts to a convex combination of two convex functions. The first one takes into account the average consensus problem while the second one concerns the estimation of the Laplacian eigenvalues.

For large graphs, numerical problems can appear. Therefore, the current formulations of the proposed Algorithms 5, 6, 7, 8 are efficient and well adapted for networks of medium size.

Finally, network topology identification refers to detecting and identifying the interested network elements and the relationship between elements in target network and represents the topology construction in an appropriate form. Therefore, identification of networks of systems becomes a growing attractive task for solving many problems in different science and engineering domains.

Theoretically, the consensus matrix $\mathbf{W}$ admits the eigenvalues decomposition $\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where $\mathbf{D} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$, with $\mathbf{\Delta}$ be diagonal matrix of $\lambda_i(\mathbf{L})$, stands for the diagonal matrix of the eigenvalues and $\mathbf{U}$ the matrix of eigenvectors. Since Laplacian spectrum $sp(\mathbf{L})$ can be achieved by means of proposed methods in Chapter 4, we proposed a method to reconstruct the network topology via the estimation of eigenvectors of the matrix $\mathbf{W}$.

6.2 Ongoing and future works

In what follows, we state some issues that are still open and worth of investigations:

1. Design of finite-time average consensus protocols:
   (a) The dependence of the convergence speed on the learning sequence is to be studied. Thus, the design of optimal learning sequences can be dig in.
   (b) Researching on the robustness of the finite-time average consensus.
2. Network robustness assessment:
   (a) Novel methods that can be applied to large-scale networks.
   (b) For indirect method, in practice, it may encounter with the ill-conditioned problem when establishing submatrices $\mathbf{Q}_i$, which leads to loose the numerical rank of these submatrices. Therefore, one can investigate in the preconditioning of these matrices to keep the accuracy of the proposed methods.

(c) Investigating in the convergence rate analysis based on the study in [25].

3. Reconstruction of the network topology:

   (a) Design a new method that deals with a Laplacian spectrum $sp(\mathbf{L})$, whose eigenvalues are not all distinct.

   (b) Assess the robustness of these methods when the data exchange are prone to imperfect communications.

Appendix

A. Khatri-Rao Product

Given two matrices $\mathbf{A} \in I\!R^{I \times F}$, $\mathbf{B} \in I\!R^{J \times F}$ with the same number of columns, the Khatri-Rao product $\mathbf{A} \odot \mathbf{B} \in I\!R^{IJ \times F}$ is defined as:

$$\mathbf{A} \odot \mathbf{B} := [\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_F \otimes \mathbf{b}_F] = \mathbf{B} \begin{pmatrix} \text{diag}_1(\mathbf{A}) \\ \vdots \\ \text{diag}_I(\mathbf{A}) \end{pmatrix}$$

where $\mathbf{a}_f$ is the f^{th} column of $\mathbf{A}$, similarly for $\mathbf{b}_f$, and $\otimes$ denotes the Kronecker product of two column vectors, whereas $\text{diag}_i(\mathbf{A})$ stands for the diagonal matrix built with the ith row of $\mathbf{A}$. The Khatri-Rao product can be viewed as a column-wise Kronecker product. Its main property, which will be used herein, is related to the vectorization operation of a given matrix:

$$\text{vec}(\mathbf{A}diag(\mathbf{d})\mathbf{B}^T) = (\mathbf{B} \odot \mathbf{A})\mathbf{d}.$$

where the $vec(.)$ operator stacks the columns of its matrix argument and $\mathbf{d} \in R^{F \times 1}$. Therefore we can deduce the following identity:

$$\mathbf{A}diag(\mathbf{d})\mathbf{b} = (\mathbf{b}^T \odot \mathbf{A})\mathbf{d}. \tag{A.0.1}$$

B. Résumé en français

Contents

B.1 Introduction

B.1.1 Contexte de la thèse

B.1.1.1 Multi-agent systems

Les systèmes multi-agents (MAS) ont reçu un intérêt croissant au cours des dernières décennies. Ils sont développés pour la demande des caractéristiques de flexibilité, robustesse, et reconfiguration qui apparaissent dans divers domaines d'applications, notamment la fabrication, la logistique, les réseaux électriques intelligents, l'automatisation des bâtiments, les opérations de secours en cas de catastrophe, les systèmes de transport intelligents, la surveillance, la surveillance de l'environnement et l'exploration des infrastructures de sécurité et protection, et cetera. Un MAS est un système composé de plusieurs agents interagissent intelligents (capteurs, les plantes, véhicules, robots, etc.) et de leur environnement, comme indiqué dans la Figure 1.

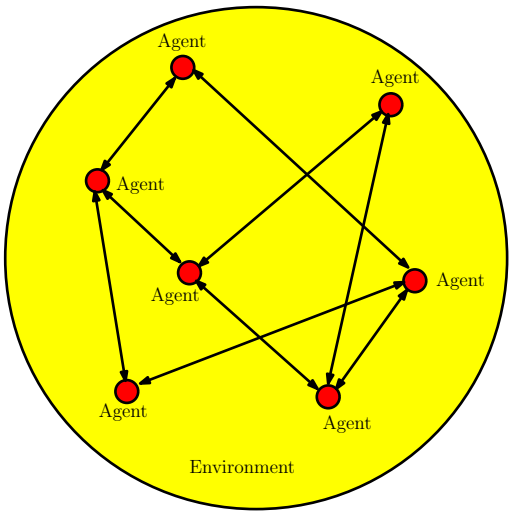


Figure 1: L'architecture générale d'un MAS.

Un agent intelligent possède les caractéristiques clés suivantes:

- *L'autonome*: la capacité de travailler sans intervention directe, jusqu'à un point défini par l'utilisateur.
- *La Réactivité*: la capacité de percevoir son environnement et réagir à ses changements, que ce soit la modification des objectifs de l'utilisateur ou des ressources disponibles.
- *Proactive*: la capacité de prendre la responsabilité de fixer ses propres objectifs.
- *La capacité sociale*: la capacité de communiquer avec les humains et d'autres agents

- *Vues locales*: Aucun agent a une vue globale du système, ou le système est trop complexe pour un agent de faire un usage pratique de ces connaissances.

Un MAS peut traiter des tâches difficiles, voire impossibles à accomplir par un agent individuel. Au cours des dernières décennies, les MASs gagnent un intérêt généralisé dans de nombreuses disciplines telles que les mathématiques, la physique, la biologie, l'informatique, les sciences sociales. Un nombre croissant de sujets de recherche des MASs comprend la coopération et la coordination, calcul distribué, contrôle automatique, les réseaux de communication sans fil, etc.

Dans la structure centralisée typique , un centre de fusion (FC) collecte toutes les mesures des agents, puis effectue les calculs finaux. Néanmoins, en raison de l'information à haut débit au FC, la congestion peut être créé. Une telle structure est vulnérable à la défaillance du FC. En outre, les besoins en matériel pour construire les communications sans fil peut engendrer d'une augmentation du coût des dispositifs et, par conséquent, un coût global du réseau plus élevé . Pour ces raisons, une structure centralisée peut devenir inefficace. Ainsi, la tendance de la recherche de la masse déplacée vers *MASs décentralisées* où l'interaction entre agents est implémenté localement sans connaissance globale. Un bon exemple est les réseaux de capteurs sans fil (WSN), qui trouvent de larges domaines d'application telles que les applications militaires (la surveillance de bataille, le monitering forces amies, de l'équipement et des munitions, etc.), les applications de l'environnement (la détection des feux de forêt, la détection de la nourriture, etc.), les applications de santé (de télésurveillance des données physiologiques humaines, etc.), domotiques, le contrôle de la formation, et cetera. Figure 2 dépeint un WSN qui collecte des données concernant la qualité de l'air, l'intensité lumineuse, le volume du son, la chaleur, les précipitations et le vent.

Figure 2: Un réseau de capteurs sans fil sur les lampadaires dans toute la ville. (Source:http://mostepicstuff.com/chicago-city-installing-smart-sensors-city-wide-to-monitor-everything)

B.1.1.2 Motivation

Il y a deux points qui attirent une grande attention de la communauté de recherche:

(A) **Sur la base de l'information locale et les interactions entre les agents, comment peuvent tous les agents parvenir à un accord?**

Ce problème est appelé *problème du consensus*, ce qui est pour concevoir un protocole de réseau basé sur les informations locales obtenus par chaque agent de telle sorte que tous les agents atteignent finalement un accord sur certaines quantités d'intérêt

Problèmes de consensus ont reçu une attention considérable de diverses communitities de recherche en raison de ses larges applications dans de nombreux domaines, y compris multi-capteurs fusion de données [64], le flocage comportement des essaims [6, 38, 59, 60], multi-véhicule de contrôle de la formation [17, 54, 62], distribué calcul [5, 10, 48], rendez-vous des problèmes [15], etc.. Plus particulièrement, les algorithmes de consensus de moyenne (c'est à dire l'accord correspond à la moyenne des valeurs initiales) sont couramment utilisés comme la composante pour plusieurs algorithmes de contrôle, estimation ou inférence distribués.

Dans la littérature récente, on peut trouver des algorithmes de consensus moyens embarqués dans le filtre de Kalman distribué, [61]; Distribué algorithme des moindres carrés, [9]; Distribué alternatif méthode des moindres carrés pour la factorisation des tenseurs, [45]; Principal Distributed Component Analysis, [49]; ou distribué entrée commune et estimation d'état, [24] pour citer quelques-uns. Néanmoins, la convergence asymptotique des algorithmes de consensus ne sont pas adaptés à ces types d'algorithmes distribués sophistiqués. La convergence faible asymptotique ne peut pas assurer l'efficacité et l'exactitude des algorithmes, ce qui peut conduire à d'autres effets inattendus. Par exemple, en ce qui concerne les réseaux de capteurs sans fil (WSNs), une réduction du nombre total d'itérations jusqu'à ce que la convergence peut conduire à une réduction de la quantité totale de consommation d'énergie du réseau, ce qui est indispensable pour garantir une durée de vie plus longue pour l'ensemble du réseau. D'autre part, les protocoles qui garantissent un temps d'exécution minimum sont plus attrayants que ceux assurant la convergence asymptotique. Dans ce but, plusieurs contributions dédiées à **consensus en temps fini** ont récemment été publiés dans la littérature. Ce qui signifie que le consensus est obtenu dans un nombre fini d'étapes.

(B) **Pouvons-nous évaluer la robustesse de ce réseau?**

En général, nous sommes entourés par des réseaux de différents types (réseaux sociaux, réseaux de capteurs, réseaux d'énergie, et cetera). Conséquent, la chose la plus importante est que ces réseaux sont robustes. Ce qui signifie que, ils peuvent

subir des dommages ou des défaillances travers. Par conséquent, la terminologie *robustesse* envahit rapidement le communauté de la recherche.

Definition 2

> *La robustesse est la capacité d'un réseau de continuer à bien fonctionner quand il est sujette à des défaillances ou des attaques, [21].*

Pour évaluer si le réseau est robuste, la mesure de la robustesse du réseau est nécessaire [21]. Il existe différentes approches proposées pour mesurer les paramètres de robustesse du réseau. Cependant, l'approche la plus populaire dans la littérature est basée sur l'analyse du graphe. **Avec les données produites durant protocole de consensus pouvons-nous déduire des mesures de robustesse du réseau?**

(C) **Pouvons-nous reconstruit la topologie du réseau?** L'identification de la topologie du réseau se réfère à détecter et identifier les éléments du réseau intéressés et la relation entre les éléments du réseau destinataire et représente la construction de la topologie sous une forme appropriée. Conséquent, l'identification des réseaux de systèmes devient une tâche de plus en plus attrayante pour résoudre de nombreux problèmes dans différents domaines scientifiques et techniques.

B.1.1.3 Objectifs de la thèse

Il est bien connu que les protocoles de consensus étant itérative, de nombreuses données sont produites. Cependant, la plupart des études se concentrent uniquement sur les propriétés de la convergence de ces algorithmes. Avec les données collectées auprès chaque itération du protocole de consensus asymptotique de manière distribuée, nous visons à:

- Concevoir des protocoles de consensus de moyenne en temps fini pour les accords subséquents.
- Évaluer la robustesse du réseau.
- Reconstruit la topologie du réseau.

B.1.2 Structure du document

Afin d'avoir une vue globale, les contributions de cette thèse peuvent être considérés triplé comme indiqué dans le schéma suivant.

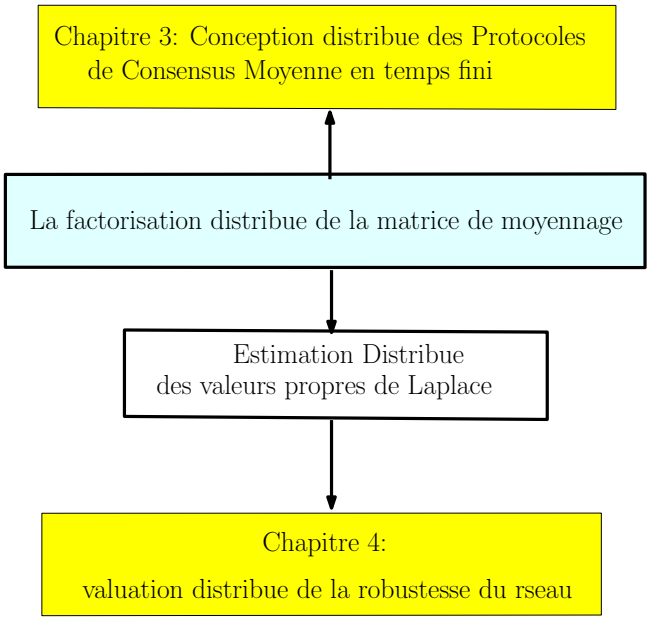


Figure 3: Schéma de la thèse

La première tâche est dédiée à la conception des protocoles de consensus de moyenne en temps fini, tandis que la seconde tâche se concentre sur l'évaluation de la robustesse du réseau. Nous présentons les grandes lignes de cette thèse comme suivant:

- Chapitre 2: Théorie des graphes.

  Dans ce chapitre, nous donnons des définitions générales sur la théorie des graphes.

- Chapitre 3: Conception distribuée des Protocoles de Consensus de Moyenne en temps fini.

  Ce chapitre est une contribution principale de cette thèse, où la conception du consensus de moyenne en temps fini est résolu de manière distribuée. Le problème est formulé comme une factorisation de la matrice moyenne $\mathbf{J}_N = \frac{\mathbf{1}\mathbf{1}^T}{N}$. La solution est obtenue en résolvant un problème d'optimisation sous contrainte d'une manière distribuée. Pour la simple compréhension, chaque itération linéaire $k, k = 0, \ldots, D$ avec D soit le nombre d'étapes est illustrée comme une couche d'un réseau (y compris $D + 1$ couches). L'idée principale est de calculer les dérivées partielles de l'erreur entre la sortie réelle et la sortie désirée, puis propager les ramener à chaque itération k pour mettre à jour les poids. Ensuite, l'algorithme proposé est basé sur la méthode de 'back-propagation' et une méthode de descente de gradient associée. L'évaluation de la méthode proposée est évaluée par les résultats numériques.

- Chapitre 4: Évaluation distribuée de la robustesse du réseau.

  Dans ce chapitre, nous utilisons des mesures qui sont le nombre d'arbres couvrants ξ et la résistance effective du graphe $\mathcal{R}$ pour évaluer la robustesse d'un réseau.

Par la nature de ces mesures, elles sont toutes les fonctions des valeurs propres du laplacien $\lambda_i(\mathbf{L})$, par conséquent, l'estimation des valeurs propres du laplacien est le but principal de cette tâche. Dans ce chapitre, au lieu d'estimer directement des valeurs propres du laplacien, nous résolvons le problème comme un problème d'optimisation sous contrainte en ce qui concerne à l'inverse des valeurs propres du laplacien distinctes différents de zéro. Par conséquent, en appliquant la méthode de descente de gradient, nous obtenons finalement l'ensemble des valeurs propres du laplacien distinctes différents de zéro. En outre, en utilisant un algorithme de programmation en nombres entiers, on peut estimer les multiplicités correspondant à ces valeurs propres. Ensuite, on en déduit l'ensemble du spectre de la matrice de Laplace associée au graphe $G(V, E)$. A partir de ce moment, les mesures de robustesse peuvent être estimés à évaluer la robustesse du réseau.

Pour la tâche d'estimation du spectre de Laplace, toutes les méthodes proposées dans ce chapitre sont divisées en deux catégories qui sont des formulations non-convexes et convexes. Selon problème d'optimisation non convexe (appelle aussi méthode directe), le problème est vue comme un problème d'optimisation sous contrainte en ce qui concerne la stepsizes α_k de protocole basé sur la Laplace matrice de consensus, dont les inverses sont égales aux valeurs propres du laplacien distinctes différents de zéro.

Résoudre le problème d'optimisation non convexe, le principal problème est la lenteur de la vitesse de convergence. Nous essayons donc d'accélérer la vitesse de l'estimation des valeurs propres du laplacien le plus vite possible. Le point principal est de tranformer le problème non convexe dans l'une convexe. Nous résolvons le problème en deux algorithmes utilisant la méthode de projection distribué, [57] et la Direction alternatif de la Méthode de multiplicateurs (ADMM), [11]. Nous considérons trois cas: la valeur moyenne finale parfaitement connue, bruyant et totalement inconnu.

- Chapitre 5: Reconstruction de la topologie du réseau. En théorie, la matrice de consensus $\mathbf{W}$ admet la décomposition des valeurs propres $\mathbf{W} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, où $\mathbf{\Lambda} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$, avec $\mathbf{\Delta}$ est matrice diagonale de $\lambda_i(\mathbf{L})$, représente la matrice diagonale des valeurs propres et $\mathbf{U}$ la matrice de vecteurs propres. Nous pouvons trouver un moyen d'obtenir $\mathbf{U}$. Puisque la spectre de Laplace $sp(\mathbf{L})$ peut être réalisée en utilisant le méthodes proposées dans le Chapitre 4, nous pouvons reconstruire la topologie du réseau.

- Chapitre 6: Conclusions générales et travaux futurs.

  Ce dernier chapitre résume les contributions de cette thèse et suggèrent également les orientations futures.

B.2 Conception distribuée des Protocoles de Consensus de Moyenne en temps fini

B.2.1 Introduction

Considérons un graphe connexe non orienté $G(V, E)$ avec $x_i(k)$ étant l'état de nœud i au étape temporelle k et définir l'état initial du réseau donné comme $\mathbf{x}(0) = [x_1(0), x_2(0), \ldots, x_N(0)]^T$ où $N = |V|$, une nouvelle méthode est proposé de concevoir les protocoles de consensus de moyenne en temps fini pour MASs ou les réseaux de capteurs sans fil (WSNs) de manière distribuée. Dans le contexte en temps discret, un schéma d'itération linéaire exprime ses avantages pour un algorithme distribué depuis dans lequel, chaque nœud met à jour à plusieurs reprises sa valeur comme une combinaison linéaire pondérée de sa propre valeur et celles de ses voisins.

$$x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), \tag{B.2.1}$$

où

$$w_{ij} = \begin{cases} 0, & \text{if } (i,j) \notin E \\ w_{ij}, & \text{if } (i,j) \in E \end{cases}$$

ou de façon équivalente sous forme de matrice:

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k). \tag{B.2.2}$$

$\mathbf{W}(k) \in \mathcal{S}_G$ où $\mathcal{S}_G$ est l'ensemble des matrices qui peuvent être factorisées comme $\mathbf{W}(k) = \mathbf{Q}(k) \circ (\mathbf{I}_N + \mathbf{A})$ où $\mathbf{Q}(k)$ supports pour une matrice carrée d'arbitraire, $\mathbf{I}_N$ étant le $N \times N$ matrice d'identité, tandis que $\circ$ désigne le produit de la matrice de Hadamard correspondant à un produit de la matrice élément par élément.

Comme présentés au Chapitre 1.2, le consensus de moyenne est atteint asymptotiquement si

$$\lim_{k \to \infty} \mathbf{x}(k) = \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}(0), \tag{B.2.3}$$

qui signifie que

$$\lim_{k \to \infty} \mathbf{W}^k = \frac{1}{N}\mathbf{1}\mathbf{1}^T = \mathbf{J}_N. \tag{B.2.4}$$

Cependant, il faut noter que pour exécuter l'algorithme de consensus de moyenne, deux étapes principales sont nécessaires: l'étape de configuration (également appelé étape de la conception) et l'étape de l'exécution. Au cours de l'étape de configuration, une tâche peut être réalisée par un algorithme d'auto-configuration au lieu de recourir à un gestionnaire de réseau. L'auto-configuration peut inclure des graphes découvertes et la

prise distribué sur certains paramètres. Par exemple, si le protocole a les poids de degré maximum (1.2.6), chaque agent calcule d'abord le nombre de ses voisins avant d'exécuter un algorithme de Max-consensus pour le calcul du degré maximum de le graphe. Dans le cas du protocole basé sur Metropolis-Hasting (1.2.7), chaque agent compare son degré avec celle de ses voisins afin de calculer les poids du protocole de consensus de moyenne. Un protocole couramment est le poids des bords constantes, ou un protocole de consensus moyen basé sur le graphe de Laplace (1.2.8), où un stepsize commun est utilisé par tous les agents. Convergence asymptotique est donc garantie si le stepsize est strictement positif et inférieur à $\frac{2}{\lambda_N}$, où λ_N représente la plus grande valeur propre de Laplace. Même à travers, il y a quelques bornes simples qui donnent des choix pour le stepsize sans nécessiter une connaissance exacte du spectre de Laplace, les agents doivent s'accorder dans un stepsize adéquate.

Au meilleur de notre connaissance, il n'y a pas de travail publié traitant avec protocoles d'auto-configuration pour le protocole de consensus de moyenne sur la base des poids des bords constantes. Dans les systèmes complexes réels, le temps d'exécution devient de plus en plus impactant. Conséquent, le but est maintenant de concevoir un algorithme de consensus de moyenne en temps fini permettant à tous les nœuds d'atteindre la valeur moyenne du consensus en un nombre fini d'étapes D pour les protocoles d'auto-configuration, c'est-à-dire,

$$\mathbf{x}(D) = \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}(0) = \mathbf{J}_N\mathbf{x}(0). \tag{B.2.5}$$

B.2.2 La solution distribuée du problème de la factorisation de la matrice

Pour un graphe connexe $G(V, E)$, soit $\mathbf{x}(0) = [x_1(0), x_2(0), \ldots, x_N(0)]^T$ le vecteur d'état du reseau $N-$agent associé à un graphe G.

Comme présenté dans le chapitre 1.2, schéma d'itération linéaire est un outil utile qui permet au système d'atteindre le consensus de moyenne en temps fini dans D étapes. Par conséquent, conformément à chaque itération, les vecteurs d'état mises à jour $\mathbf{x}(k), k = 1, \ldots, D-1$ sont assignés à être des couches consécutives respectivement. Enfin, la dernière itération est dédiée à des valeurs output $\mathbf{x}(D)$.

Les matrices de poids $\mathbf{W}(k), k = 1, \ldots, D$ qui ont les entrées w_{ij}^k sont correspondant aux liens entre les nœuds de deux couches consécutives à étape temporelle k.

Remark 13

Les matrices de poids $\mathbf{W}(k) \in \mathcal{S}_G$ est associées à un graphe G. Par conséquent, les

entrées $w_{ij}(k)$ sont conçus suivant la règle définie comme:

$$\boldsymbol{W}_{ij}(k) = \begin{cases} w_{ij}^k, & \text{if } j \in N_i \\ 0, & \text{otherwise} \end{cases}$$

Afin de simplifier, à partir de maintenant, nous noterons $\mathbf{W}(k)$ comme $\mathbf{W}_k$.

Le schéma des itérations linéaires dans l'espace et le temps peut être considéré comme un réseau neuronal multicouche. La sélection des poids peut être analysée à travers le portée de la formation. En utilisant l'ensemble des P séquences d'apprentissage, le mécanisme est divisé en deux étapes principales, à savoir, étape en avant et étape en arrière. *L'idée principale est de calculer les dérivées partielles de l'erreur entre le output réell et le output désiré, puis propager les ramener à chaque couche du réseau afin de mettre à jour les poids.* L'objectif est de réduire le critère d'erreur E efficacement.

Soit $\{x_{i,p}(0), y_{i,p}\}$, $i = 1, \cdots, N$, $p = 1, \cdots, P$, le input-output paires définissant les séquences d'apprentissage, avec $y_{i,p} = y_p = \frac{1}{N}\sum_{i=1}^{N} x_{i,p}(0)$. Notre objectif est d'estimer les matrices de facteurs $\mathbf{W}_k$, $k = 1, \cdots, D$, en minimisant l'erreur quadratique

$$E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\sum_{i=1}^{N}\sum_{p=1}^{P}(x_{i,p}(D) - y_p)^2, \tag{B.2.6}$$

avec $x_{i,p}(k) = \sum_{j \in N_i \cup \{i\}} w_{ij}^k x_{j,p}(k-1)$, w_{ij}^k étant les entrées de matrices $\mathbf{W}_k \in \mathcal{S}_G$. Nous pouvons réécrire la fonction de coût (B.2.6) comme

$$E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\left\| \prod_{k=D}^{1} \mathbf{W}_k \mathbf{X}(0) - \mathbf{Y} \right\|_F^2, \; s.t \; \mathbf{W}_k \in \mathcal{S}_G, \tag{B.2.7}$$

où $\|.\|_F$ supports pour la norme de Frobenius, $\mathbf{Y} = \mathbf{J}_N\mathbf{X}(0)$, $\mathbf{Y}$ et $\mathbf{X}(0)$ étant $N \times P$ matrices avec $y_{i,p}$ and $x_{i,p}$ comme entrées, respectivement.

Nous supposons que $\mathbf{X}(0)\mathbf{X}(0)^T = \mathbf{I}_N$ ce qui signifie que le vecteur d'entrée est orthogonal. Par exemple, les vecteurs de la base canonique de $\mathbb{R}^N$ peuvent être utilisés comme 'inputs'. Par conséquent, nous pouvons noter que $E(\mathbf{W}_1, \cdots, \mathbf{W}_D) = \frac{1}{2}\left\| \prod_{k=D}^{1} \mathbf{W}_k - \mathbf{J}_N \right\|_F^2$, ce qui signifie que la minimisation (B.2.7) est équivalent à résoudre le problème de la factorisation (3.2.5):

$$\{\mathbf{W}_k^*\}|_{k=1,\ldots,D} = \arg \min_{\{\mathbf{W}_k\}_{t=1,\ldots,D}} \frac{1}{2}\sum_{p=1}^{P} tr\left(\epsilon_p(\mathbf{W})\epsilon_p^T(\mathbf{W})\right), \tag{B.2.8}$$

avec

$$\epsilon_p(\mathbf{W}) = \prod_{k=D}^{1} \mathbf{W}_k \mathbf{x}_p(0) - \mathbf{y}_p, \tag{B.2.9}$$

où $tr(.)$ désigne l'opérateur de trace et $\mathbf{y}_p = \mathbf{J}_N \mathbf{x}_p(0)$.

En dénotant $E_p(\mathbf{W}) = \frac{1}{2} tr\left(\epsilon_p(\mathbf{W})\epsilon_p^T(\mathbf{W})\right)$, la solution de ce problème d'optimisation peut alors être obtenue par itération au moyen d'une méthode de descente de gradient:

$$\mathbf{W}_k := \mathbf{W}_k - \alpha \sum_{p=1}^{P} \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_k}$$

ou une méthode de gradient stochastique:

$$\mathbf{W}_k := \mathbf{W}_k - \alpha \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_k},$$

où le gradient de la fonction coût est approximé par le gradient à une séquence d'entrée-sortie unique. Dans ce qui suit, nous avons recours à une méthode de gradient stochastique. Dans ce but, nous affirmons d'abord le lemme technique suivant:

Lemma 6

Les dérivés de la fonction de coût $E_p(\boldsymbol{W}) = \frac{1}{2} tr\left(\epsilon_p(\boldsymbol{W})\epsilon_p^T(\boldsymbol{W})\right)$ avec $\epsilon_p(\boldsymbol{W})$ défini dans la (B.2.9) peuvent être calculées comme suit:

$$\frac{\partial E_p(\boldsymbol{W})}{\partial \boldsymbol{W}_k} = \boldsymbol{\delta}_{k,p} \mathbf{x}_p^T(k-1), k = 1, \ldots, D \tag{B.2.10}$$

$$\frac{\partial E_p(\boldsymbol{W})}{\partial \boldsymbol{W}_D} = \boldsymbol{\delta}_{D,p} \mathbf{x}_p^T(D-1) \tag{B.2.11}$$

où $\boldsymbol{\delta}_{D,p} = \mathbf{x}_p(D) - \bar{\mathbf{x}}_p$ est la différence entre la sortie réelle et la sortie désirée avec $\bar{\mathbf{x}}_p = \mathbf{y_p} = \frac{1}{N} \sum_{i=1}^{N} x_{i,p}(0)$; et

$$\boldsymbol{\delta}_{k-1,p} = \boldsymbol{W}_k^T \boldsymbol{\delta}_{k,p}, k = 1, \ldots, D. \tag{B.2.12}$$

En appliquant les résultats de Lemme 6, le schéma de mise à jour de l'algorithme d'optimisation est la suivante:

$$\mathbf{W}_k[t+1] = \mathbf{W}_k[t] - \alpha \frac{\partial E_{p(t)}(\mathbf{W})}{\partial \mathbf{W}_k} = \mathbf{W}_k[t] - \alpha \boldsymbol{\delta}_{k,p(t)} \mathbf{x}_{p(t)}^T(k-1), \tag{B.2.13}$$

où $p(t) \in \{1, 2, \cdots, P\}$, et t supports pour les t^{th} itération du processus d'optimisation. Ensuite, nous projetons les poids mis à jour sur $\mathcal{S}_G$ qui donne:

$$\mathbf{W}_k[t+1] = \mathbf{W}_k[t] - \alpha \frac{\partial E_{p(t)}(\mathbf{W})}{\partial \mathbf{W}_k}$$

$$= \mathbf{W}_k[t] - \alpha \left(\left[\boldsymbol{\delta}_{k,p(t)} \mathbf{x}^T_{p(t)}(k-1) \right] \circ (\mathbf{I}_N + \mathbf{A}) \right), \qquad \text{(B.2.14)}$$

L'algorithme de descente de gradient (B.2.14) aagit en alternant les étapes suivantes: la séquence d'apprentissage est d'abord propagé vers l'avant. Ensuite, l'erreur entre la sortie 'output' ciblée et $\mathbf{x}(D)$ est calculée et ensuite propagé.

Proposition 6

Soit $(\boldsymbol{W}^*_1, \ldots, \boldsymbol{W}^*_D)$ *be a local solution of* $E(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_D)$, *puis*

$$\prod_{k=D}^{1} \boldsymbol{W}^*_k = \boldsymbol{J}_N$$

.

Ce mécanisme est similaire à l'algorithme de rétro-propagation de gradient. Sa convergence a été bien étudié dans la littérature, voir [14] et [50] par exemple. Convergence vers un minimum local est garanti si le step-size α est choisi de manière appropriée dans l'intervalle $(0 < \alpha < 1)$. Si le step-size α est trop grande, l'erreur E tend à osciller, n'atteint jamais un minimum. En revanche, trop petite α empêche l'optimisation de faire des progrès raisonnables. Conséquent, le choix d'un appropriée α est aussi une difficulté. Une telle step-size influe également sur la vitesse de convergence de l'algorithme. Une astuce pour accélérer la convergence est d'ajouter un terme de régularisation dans la fonction de coût à minimiser.

$$\{\mathbf{W}^*_k\}|_{k=1,\ldots,D} = \arg \min_{\{\mathbf{W}_k\}_{k=1,\ldots,D}} \sum_{p=1}^{P} E_{p(t)}(\mathbf{W}) + \frac{1}{2} \sum_{k=1}^{D} \beta \|\mathbf{W}_k[t] - \mathbf{W}_k[t-1]\|^2.$$

En minimisant une fonction de coût, la mise à jour équation est donnée par:

$$\mathbf{W}_k[t+1] = \mathbf{W}_k[t] - \alpha \left(\left[\boldsymbol{\delta}_{k,p(t)} \mathbf{x}^T_{p(t)}(k-1) \right] \circ (\mathbf{I}_N + \mathbf{A}) \right) + \beta(\mathbf{W}_k[t] - \mathbf{W}_k[t-1]).$$

Élément par élément, chaque agent met à jour les entrées de la façon suivante:

$$w^k_{ij} := w^k_{ij} - \alpha(\delta_{i,k} x_j(k-1)\hat{a}_{ij} + \beta(w^k_{ij} - w^{(k-1)}_{ij}),$$

où:

- $\delta_{i,k}$ est le i-eme entrée de $\boldsymbol{\delta}_k$ et $x_j(k-1)$ le j-eme entrée de $\mathbf{x}(k-1)$. Ici, $\hat{a}_{ij}$-l'entrée de la matrice $(\mathbf{I}_N + \mathbf{A})$ est défini comme:

$$\hat{a}_{ij} = \begin{cases} a_{ij} & \text{if } i \neq j. \\ 1 & \text{if } i = j. \end{cases}$$

- α et β sont le taux d'apprentissage et le taux de l'élan respectivement. Ils peuvent être choisis comme constantes ou des paramètres variables. Dans notre étude, le taux d'apprentissage $\alpha[t]$ varie à chaque étape d'itération d'optimisation t tandis que le taux de l'élan est fixé.

Pour éclairer le problème, Figure 4 montre le mécanisme de l'algorithme proposé. Dans le détail, pour chaque séquence d'apprentissage, l'algorithme exécute d'abord l'étape en avant, où chaque nœud met à jour à plusieurs reprises sa valeur. À la fin de cette étape, chaque couche stocke les valeurs mises à jour, puis les dérivées partielles de l'erreur δ_D entre $\mathbf{x}(D)$ et les valeurs moyennes souhaitées de consensus $\bar{\mathbf{x}}$ sont estimés. L'étape en arrière commence à propager retour les dérivés partiels estimés δ_D à chaque couche jusqu'à ce que la couche d'entrée reçoit le δ_1. Maintenant, tous les poids peuvent être mis à jour. Le régime se poursuit avec la séquence d'apprentissage suivante. Le mécanisme est répété jusqu'à ce que les critères d'erreur E satisfait le seuil θ, qui est un petit nombre.

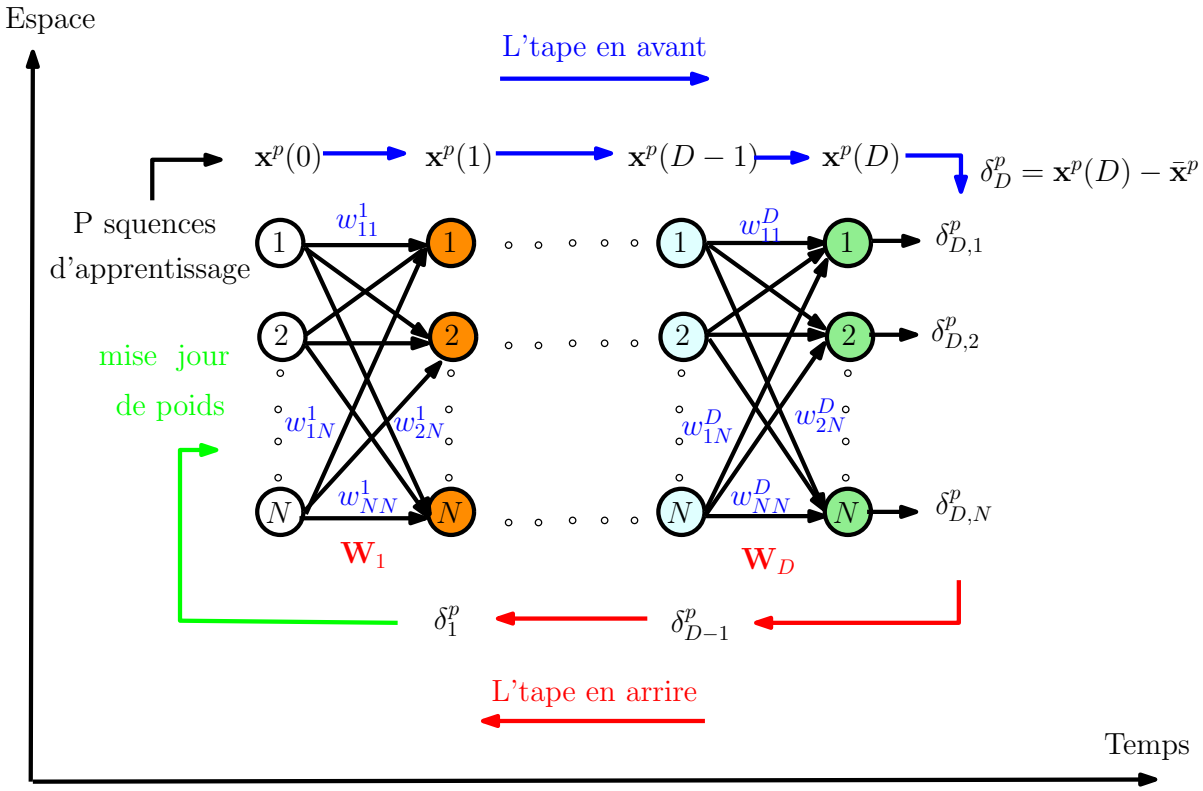


Figure 4: Mécanisme de la méthode proposée basée sur le concept de rétropropagation.

L'algorithme correspondante est alors décrit comme dans l'algorithm 3.

B.3 L'évaluation distribuée de la robustesse du réseau

B.3.1 Introduction

L'efficacité d'un réseau est évaluée par sa fonctionnalité et sa robustesse. Selon la propriété plus tard, certaines questions peuvent se poser. Par exemple, s'il y a un événement accidentel, comment réagit-il le réseau? Sera le réseau survécu? Afin de répondre à ces questions, l'étude sur la robustesse des réseaux de plus en plus attiré une attention de la communauté scientifique.

Dans la littérature, il y a beaucoup de mesures de robustesse proposé (voir la Section 1.3), [20, 21]. Par exemple, la première mesure est basée sur la connexité du graphe représentant le réseau [16]. Par les autres mots, le réseau est robuste s'il existe la présence des chemins alternatif, qui assurent la possibilité de communication en dépit des dommages [16]. Il y a deux concepts classiques de connectivité pour un graphe qui peut être utilisé pour modéliser la robustesse du réseau qui sont les sommet et bord connectivités. Le sommet (bord) connectivité $\mathcal{K}_v(\mathcal{K}_e)$ est le nombre minimal de sommets (de bords) d'être retiré afin de déconnecter le graphe donné (voir Subsection 1.3.1). Par cette approche analytique, afin d'en déduire l'indice de la robustesse, nous devons enlever chaque bord pour vérifier la connectivité du graphe. Même si cette approche est simple, c'est aussi un fardeau si nous analysons des réseaux plus grands et plus complexes raisonnables.

Dans [72], les auteurs ont proposé le concept de la connectivité naturelle comme une mesure spectrale de robustesse dans les réseaux complexes. La connectivité naturel est exprimée sous forme mathématique comme la moyenne des valeurs propres de la matrice d'adjacence du graphe représentant la topologie du réseau. L'avantage de cette méthode est que la mesure proposée fonctionne dans les deux réseaux connectés et déconnectés. D'autre part, le spectre de Laplace $sp(\mathbf{L})$ peut être utilisé pour calculer les indices de robustesse qui évalue la performance du réseau donné. Par exemple, la deuxième plus petite valeur propre du spectre de Laplace $\lambda_2(\mathbf{L})$, défini également comme la connectivité algébrique [28], est bien connu comme un paramètre critique que influence sur la performance et robustesse des systèmes dynamiques en raison de son rôle principal dans la connectivité du graphe [20]. Par conséquent, il y a beaucoup de recherches sur cette connectivité algébrique dans la littérature [3, 51, 76]. En outre, il existe des mesures remarquables pour obtenir les indices de robustesse souhaités: la résistance effective du graphe $\mathcal{R}$ et Nombre d'arbres couvrants ξ (voir Section 1.3), qui sont basées sur la tout le spectre de Laplace différents de zéro.

Nous proposons des méthodes pour estimer la résistance effective du graphe $\mathcal{R}$ et Nombre d'arbres couvrants ξ. Figure 5 montre que l'estimation distribué de la robustesse du

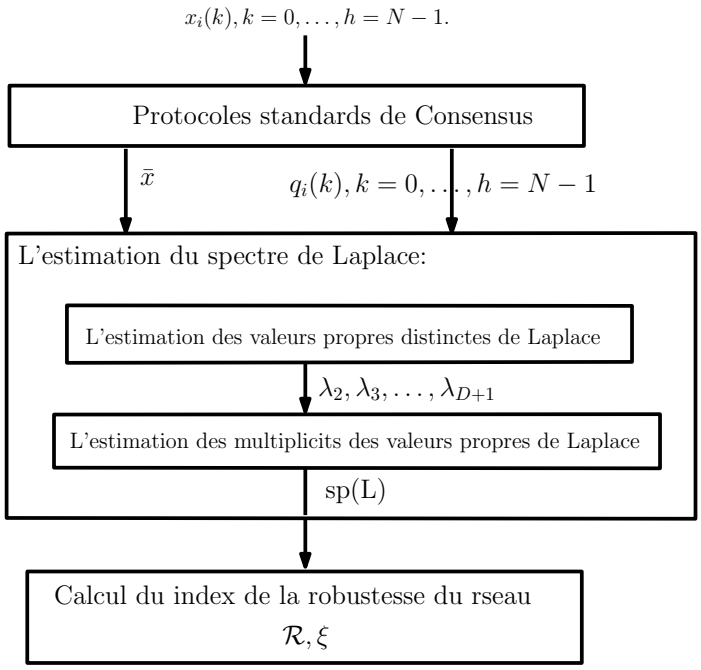


Figure 5: Schéma d'estimation distribuée de la robustesse du réseau.

réseau est divisé en deux phases principales y compris le spectre de Laplace $sp(\mathbf{L})$, la computation de la résistance effective du graphe et Nombre d'arbres couvrants $(\mathcal{R}, \xi)$. Tout d'abord, nous proposons des méthodes basées sur le consensus pour estimer tout le spectre de la matrice de Laplace de manière distribuée. Principalement, nous calculons la valeur moyenne de consensus $\bar{x}$ en utilisant les protocoles standards de consensus (Algorithme 1) ou algorithme décentralisée de calcul de valeur de consensus en temps minimum [78]. Ensuite, l'estimation du spectre de Laplace peut être divisée en deux étapes qui sont l'estimation des valeurs propres du laplacien distinctes, puis l'évaluation des multiplicités correspondantes. Enfin, à partir de $sp(\mathbf{L})$ estimé les indices de robustesse peuvent être calculées. Notre objectif est donc sur l'estimation du spectre de Laplace $sp(\mathbf{L})$.

B.3.2 Les solutions distribuées de la robustesse du réseau

B.3.2.1 L'estimation Distribuée des valeurs propres du laplacien

B.3.2.2 Formulation du problème

Considérons un réseau modélisé par un graphe non orienté connecté $G(V, E)$. Son état $\mathbf{x}(k) = [x_1(k), x_2(k), \ldots, x_N(k)]^T$ évolue comme suit:

$$\mathbf{x}(k+1) = (\mathbf{I}_N - \alpha\mathbf{L})\mathbf{x}(k) = (\mathbf{I}_N - \alpha\mathbf{L})^k\mathbf{x}(0), \tag{B.3.1}$$

où α est choisi tel que $0 < \alpha < \frac{1}{d_{max}}$ pour assurer la convergence asymptotique [74]. De l'état $\mathbf{x}(k)$, on peut définir $\mathbf{q}(k) = \mathbf{L}^k\mathbf{x}(0)$ comme:

$$\mathbf{q}(k) = (q_{1,k}, \dots, q_{N,k})^T = \frac{1}{(-\alpha)^k}\left(\mathbf{x}(k) - \sum_{i=0}^{k-1}\binom{k}{i}(-\alpha)^i\mathbf{q}(i)\right), \tag{B.3.2}$$

avec $\mathbf{q}(0) = \mathbf{x}(0)$.

1. Hypothèse 1: Le nombre N de nœuds est connue.
2. Hypothèse 2: Chaque nœud i sauve sa valeur initiale $x_i(0)$ et les valeurs consécutives $q_{i,k}, k = 0, \dots, h$.

Le problème étude peut donc être formulé comme suit:

Étant donné une paire d'entrée-sortie arbitraire $\{\mathbf{x}(0), \bar{\mathbf{x}} = \bar{x}\mathbf{1}\}$, où $\bar{\mathbf{x}} = \frac{\mathbf{1}\mathbf{1}^T}{N}\mathbf{x}(0)$, *et les observations intermédiaires* $\boldsymbol{q}(k), k = 0, \dots, h$, *estimer de manière distribuée les valeurs propres du laplacien du graphe représentant le réseau.*

Cependant, au lieu de calculer directement les valeurs propres du laplacien, nous allons calculer l'inverse des valeurs propres du laplacien. En outre, puisque le nombre D de valeurs propres du laplacien distinctes différents de zéro est inconnu, la factorisation sera ré-paramétré. Énonçons d'abord la proposition suivante:

Proposition 7

Considérons un graphe connexe avec la matrice de Laplace $\mathbf{L}$ *et ses valeurs propres distinctes différents de zéro* $\lambda_2, \cdots, \lambda_{D+1}$. *Supposons que le vecteur des valeurs locales* $\mathbf{x}(0)$ *n'est pas orthogonal à n'importe quel vecteur propre de* $\mathbf{L}$, *la fonction de coût*

$$E(\boldsymbol{\alpha}) = \|\mathbf{x}(h) - \bar{\mathbf{x}}\|^2 = \left\|\prod_{k=h}^{1}(\boldsymbol{I}_N - \alpha_k\boldsymbol{L})\mathbf{x}(0) - \bar{\mathbf{x}}\right\|^2, \tag{B.3.3}$$

avec $\boldsymbol{\alpha} = (\alpha_1 \cdots \alpha_h)^T$, $h \geq D$, *est minimale si et seulement si* $\{1/\lambda_2, \cdots, 1/\lambda_{D+1}\} \subseteq \{\alpha_1, \alpha_2, \cdots, \alpha_h\}$.

De la Proposition 7, nous pouvons conclure que l'ensemble $\mathbf{S}_1 = \{\alpha_k\}, k = 1, \dots, h$, des solutions de (B.3.3) contient l'inverse des valeurs propres du laplacien $\mathbf{S}_2 = \{\frac{1}{\lambda_i}\}, i = 2, \dots, D+1$: $\mathbf{S}_2 \subset \mathbf{S}_1$. Par conséquent, la procédure d'estimation doit être divisée en deux parties. Nous recevons d'abord $\mathbf{S}_1$ en résolvant (B.3.3) de manière distribuée, nous mettons à l'ensemble $\mathbf{S}_2$ de $\mathbf{S}_1$ et déduisons la valeurs propres du laplacien localement.

Nous savons que si $\{\alpha_k\}_{k=1}^h$ contient D inverses des valeurs propres du laplacien différent de zéro, puis

$$\bar{\mathbf{x}} = \prod_{k=1}^{h}(\mathbf{I} - \alpha_k\mathbf{L})\mathbf{x}(0),$$

ou de façon équivalente,

$$\bar{\mathbf{x}} = \sum_{k=0}^{h} c_k \mathbf{L}^k \mathbf{x}(0) = \sum_{k=0}^{h} c_k \mathbf{q}(k) = \mathbf{Q}\mathbf{c}, \tag{B.3.4}$$

où $\mathbf{Q} = [\mathbf{q}(0)\ \mathbf{q}(1)\ \cdots \mathbf{q}(h)] \in I\!R^{N\times(h+1)}$ and $\mathbf{c} = [c_0\ c_1\ \cdots\ c_h]^T \in I\!R^{(h+1)\times 1}$, le coefficient c_i étant défini comme:

$$c_k = \begin{cases} 1, & \text{if } k = 0. \\ (-1)^k \sum_{i<j<\cdots<k} \alpha_i \alpha_j \cdots \alpha_k, & \text{if } k = 1, \ldots, h-1. \\ (-1)^h \prod_{k=1}^{h} \alpha_t, & \text{if } k = h. \end{cases} \tag{B.3.5}$$

Il est évident que α_k sont les racines du polynôme avec des coefficients c_k. Et, $\mathbf{c}$ a un motif de signe alternatif.

Nous pouvons affirmer la proposition suivante qui est un corollaire de Proposition 7:

Proposition 8

Soit $\mathcal{P}(\boldsymbol{c},:)$ le polynôme de $h^{eme} degravecdescoefficients \mathbf{c}=[c_0\ c_1\ \ldots\ c_h]^T \in I\!R^{(h+1)\times 1}$ construit de l'ensemble de stepsize $\boldsymbol{S}_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_h\}$ en utilisant (B.3.5). Pour ne importe quel nœud i et des mesures intermédiaires $\boldsymbol{q}_i = [q_{i,0}, q_{i,1}, \ldots, q_{i,h}]^T \in I\!R^{(h+1)\times 1}$ du protocole de consensus (B.3.1), nous obtenons

$$\bar{x} = \mathcal{P}(\boldsymbol{c}, \boldsymbol{q}_i) = \boldsymbol{q}_i^T \boldsymbol{c} \tag{B.3.6}$$

si et seulement si les D inverses des valeurs propres du laplacien distinctes sont contenus dans $\boldsymbol{S}_1$.

Étant donné une limite supérieure h pour le nombre D de valeurs propres du laplacien distinctes, l'idée est de minimiser (B.3.3) puis de réduire étape par étape l'ensemble obtenu des coefficients c_k construite de $\mathbf{S}_1$ jusqu'à ce qu'il soit minime. Après tout, l'ensemble des inverses des valeurs propres du laplacien distinctes $\mathbf{S}_2$ est atteint. est atteint. Afin d'obtenir l'ensemble de spectre de Laplace $sp(\mathbf{L})$, les multiplicités correspondantes $m_i, i = 1, \ldots, D+1$ des valeurs propres du laplacien distinctes $\Lambda = \{\lambda_1, \ldots, \lambda_{D+1}\}$ doit être estimée (il sera considéré dans la Section B.3.2.6).

Afin de mieux comprendre la structure de ce Chapitre, Figure 6 illustre sa construction en montrant la façon d'estimer la robustesse du réseau.

En fait, pour estimation distribué des valeurs propres du laplacien, trois cas ont été profondément considéré:

1. Le cas de la parfaite connaissance de la valeur moyenne du consensus $\bar{x}$,

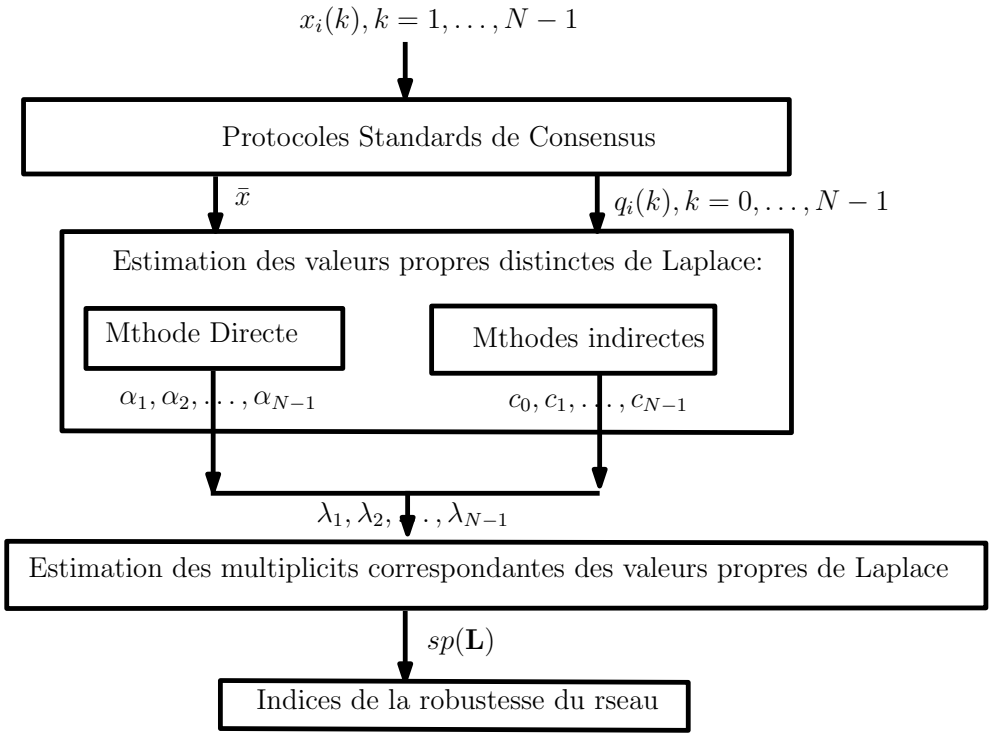


Figure 6: Mechanism of Network Robustness Estimation

2. Le cas de la connaissance imparfaite de la valeur moyenne du consensus $\bar{x}$,

3. Le cas de la valeur moyenne de consensus inconnue $\bar{x}$.

B.3.2.3 La factorisation basée sur Laplace de la matrice moyenne avec une connaissance parfaite de $\bar{x}$

Étant donné une paire d'entrée-sortie arbitraire $\{\mathbf{x}(0), \bar{\mathbf{x}}\}$ et les observations intermédiaires $\mathbf{q}(k), k = 0, 1, \ldots, h = N-1$, on peut minimiser la fonction de coût (B.3.3) pour obtenir l'ensemble des stepsizes $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{N-1})$, dont les inverses sont les valeurs propres du laplacien $\{\lambda_1, \ldots, \lambda_{N-1}\}$.

De cette façon, la méthode directe proposée en ce qui concerne les stepsizes $\boldsymbol{\alpha}$ consiste à résoudre une optimisation non convexe (B.3.3) par la méthode des multiplicateurs de Lagrange.

Méthode 1: La factorisation Directe distribuée de la matrice moyenne.

(B.3.3) montre que α_k sont des paramètres globaux. Pour l'exécution distributive la factorisation de la matrice comme des facteurs de matrices de consensus basés sur Laplace,

nous pouvons d'abord noter que la fonction de coût (B.3.3) peut être réécrite comme:

$$E(\boldsymbol{\alpha}) = \|\mathbf{x}(h) - \bar{\mathbf{x}}\|^2 = \| \prod_{k=h}^{1} (\mathbf{I}_N - diag(\boldsymbol{\alpha}_k)\mathbf{L})\mathbf{x}(0) - \bar{\mathbf{x}}\|^2, \tag{B.3.7}$$

avec $\boldsymbol{\alpha}_k = [\alpha_{1,k}, \alpha_{2,k}, \ldots, \alpha_{N,k}]^T = \alpha_k \mathbf{1}, k = 1, 2, \ldots, h = N - 1$.

L'idée de la méthode proposée est de minimiser le désaccord entre voisins sur la valeur de α_k tout en veillant que la factorisation de la matrice moyenne est obtenue. Une telle factorisation est évaluée en contraignant les valeurs des nœuds après h itérations de l'algorithme de consensus pour être égale à la moyenne des valeurs initiales:

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in I\!R^{N\times 1}, k=1,2,\ldots,h} \quad & \frac{1}{2} \sum_{k=1}^{h} \sum_{i \in V} \sum_{j \in N_i} (\alpha_{j,k} - \alpha_{i,k})^2, \\ \text{subject to} \quad & \mathbf{x}(h) = \bar{\mathbf{x}} \end{aligned} \tag{B.3.8}$$

or equivalently,

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in I\!R^{N\times 1}} \quad & \frac{1}{2} \sum_{k=1}^{h} \boldsymbol{\alpha}_k^T \mathbf{L} \boldsymbol{\alpha}_k. \\ \text{subject to} \quad & \mathbf{x}(h) = \bar{\mathbf{x}} \end{aligned} \tag{B.3.9}$$

Remark 14

Importe quelle solution de (B.3.9) contient l'inverse des valeurs propres du laplacien selon à Proposition 7.

Le problème d'optimisation sous contrainte (B.3.9) peut ensuite être réalisée comme un problème d'optimisation non contrainte au moyen d'une méthode de Lagrange avec une fonction de Lagrange définis comme suit:

$$H(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_h, \mathbf{y}) = \frac{1}{2} \sum_{k=1}^{h} \boldsymbol{\alpha}_k^T \mathbf{L} \boldsymbol{\alpha}_k + \mathbf{y}^T (\mathbf{x}(h) - \bar{\mathbf{x}}) \tag{B.3.10}$$

où $\mathbf{y} \in I\!R^{N\times 1}$ est le vecteur des multiplicateurs de Lagrange.

La minimisation de la fonction de Lagrange (B.3.10) peut être considéré comme h des problèmes de consensus à résoudre simultanément avec une contrainte qui donne une sorte de référence externe à suivre.

Pour éviter tout malentendu entre l'étape temporelle k et l'itération de l'optimisation t, nous notons $\mathbf{x}(k), \mathbf{W}(k)$ comme $\mathbf{x}_k, \mathbf{W}_k$ respectivement.

Afin de tirer l'algorithme distribué correspondant, nous énonçons maintenant le lemme technique suivant:

Lemma 7

Les dérivés de la fonction de coût $H(\boldsymbol{\alpha}_k, \mathbf{y})$ défini en (B.3.9) peut être calculée comme suit:

$$\frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k} = \boldsymbol{L}\boldsymbol{\alpha}_k - diag^{-1}(\boldsymbol{\alpha}_k) diag(\mathbf{x}_{k-1} - \mathbf{x}_k)\boldsymbol{\delta}_k, \tag{B.3.11}$$

où

$$\boldsymbol{\delta}_h = \mathbf{y} \text{ and } \boldsymbol{\delta}_{k-1} = \boldsymbol{W}_k \boldsymbol{\delta}_k, \quad k = 1, \ldots, h. \tag{B.3.12}$$

En appliquant les résultats de Lemme 7, le régime de la mise à jour de l'algorithme d'optimisation est la suivante:

$$\begin{aligned} \boldsymbol{\alpha}_k[t+1] &= \boldsymbol{\alpha}_k[t] - \beta \frac{\partial H(\boldsymbol{\alpha}_k, \mathbf{y})}{\partial \boldsymbol{\alpha}_k[t]} \\ &= (\mathbf{I}_N - \beta \mathbf{L})\boldsymbol{\alpha}_k[t] + \beta \text{diag}^{-1}(\boldsymbol{\alpha}_k)\text{diag}(\mathbf{x}_{k-1}[t] - \mathbf{x}_k[t])\boldsymbol{\delta}_k) \qquad \text{(B.3.13)} \\ \mathbf{y}[t+1] &= \mathbf{y}[t] + \mu(\mathbf{x}_h[t] - \bar{\mathbf{x}}). \qquad \text{(B.3.14)} \end{aligned}$$

L'algorithme correspondante est alors décrit comme dans l'algorithm 4.

Afin d'accélérer la vitesse de convergence, on peut re-paramétrer le problème (B.3.1) au moyen de l'équation (B.3.4).

$$\bar{\mathbf{x}} = \sum_{k=0}^{h} c_k \mathbf{L}^k \mathbf{x}(0) = \sum_{k=0}^{h} c_k \mathbf{q}(k) = \mathbf{Q}\mathbf{c}.$$

Au lieu de la minimisation les step-sizes α_k, la nouvelle optimisation les coefficients du polynôme $c_k, k = 0, \ldots, h$ peut être appliquée. Cela s'appelle factorisation distribué indirect de la matrice moyenne.

Méthode 2: La factorisation indirecte distribuée de la matrice moyenne.

Le but de cette méthode est de trouver le polynôme coefficents c_k de manière distribuée en résolvant le problème:

$$\min_{\mathbf{c}} \|\bar{\mathbf{x}} - \mathbf{Q}\mathbf{c}\|^2,$$

avec $\mathbf{Q} = [\mathbf{q}(0)\ \mathbf{q}(1)\ \cdots\ \mathbf{q}(h)] \in I\!R^{N \times (h+1)}$.

Toutes les conditions initiales $x_i(0)$ ne devrait pas être vecteurs propres de la matrice de Laplace $\mathbf{L}$ et ne pas être identiques, qui signifie que $\mathbf{x(0)} \neq \delta\mathbf{1}, \delta \in I\!R$, sinon toutes les colonnes de $\mathbf{Q}$ sont nuls sauf la première. Puisque la condition initiale est générée de façon aléatoire, presque sûrement, il ne peut pas être un vecteur propre de la matrice de Laplace $\mathbf{L}$.

Lemma 8

Considérons un graphe non orienté connecté $G(V, E)$ avec le spectre de Laplace

$sp(\boldsymbol{L}) = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$. Ici, $sp(\boldsymbol{L})$ a $D+1$ valeurs propres du laplacien distinctes. Si $\mathbf{x}(0)$ n'est pas un vecteur propre de la matrice de Laplace $\boldsymbol{L}$, alors la matrice

$$\boldsymbol{Q} = \boldsymbol{U} diag(\boldsymbol{U}^T \mathbf{x}(0)) \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^h \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^h \\ & \cdots\cdots & & & \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^h \end{pmatrix}$$

a rang $min(D+1, N)$. Où $\boldsymbol{U}$ est la matrice des vecteurs propres de Laplace.

Remark 15

$rank(\boldsymbol{Q}) < N$ si existe $\lambda_i \in sp(\boldsymbol{L})$ avec multiplicité supérieure à 1, $\forall \mathbf{x}(0)$.

D'une manière centralisée, afin de trouver $\mathbf{c}$ de l'équation $\mathbf{Qc} = \bar{\mathbf{x}}$, $\mathbf{Q}$ doit être une matrice de rang plein . Ensuite,

$$\mathbf{c} = \mathbf{Q}^{-1}\bar{\mathbf{x}}$$

En contraste, si $\mathbf{Q}$ n'est pas une matrice de rang plein, alors il y a plusieurs solutions de $\mathbf{c}$. Pour faire face à cette situation, nous utilisons la solution des moindres carrés permettant de trouver la plus courante solution S_1 à l'intérieur de L'ensemble des solutions $(So_1, So_2, \ldots, So_m)$ de $\mathbf{c}$ comme décrit dans Figure 7.

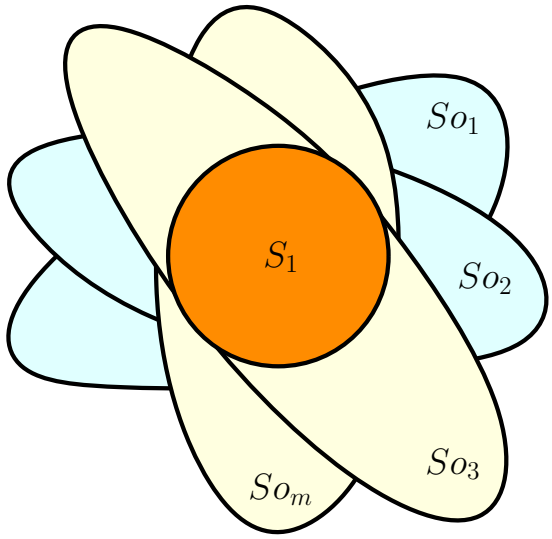


Figure 7: Plusieurs solutions de $\mathbf{c}$ dans le cas d'une matrice de rang que n'est pas plein $\mathbf{Q}$.

Le problème devient:

$$\min_{\mathbf{c}} \|\mathbf{c}\|^2 \text{ subject to } \mathbf{Qc} = \bar{\mathbf{x}}$$

Ici, $\mathbf{Q}$ est une matrice de la ligne rang plein.

Afin de résoudre ce problème de manière distribuée, nous résolvons le problème localement selon à $\mathbf{c}_i, i = 1, \ldots, N$. En désignant par $\mathbf{c}_i$ la version locale de $\mathbf{c}$ et $\mathbf{Q}_i \in I\!R^{(d_i+1)\times(h+1)}$ la sous-matrice de $\mathbf{Q}$ constitué avec les lignes associées à nœud i et ses d_i voisins, la solution du problème précédent peut être obtenue d'une manière distribuée en résolvant

$$\min_{\mathbf{c}_i} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 \quad s.t. \quad \mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}, \ i = 1, \cdots, N, \tag{B.3.15}$$

pourvu que $\mathbf{Q}_i$ est rangée rang plein.

Appelons $\mathbf{Se}_i$ une matrice de sélection rangée qui concerne au nœud i. $\mathbf{Se}$ est une matrice, où $Se_{jj} = 1$ si $j \in (i \cup N_i)$. $\mathbf{Q}_i$ peut être exprimée comme:

$$\mathbf{Q}_i = \mathbf{Se}_i \mathbf{Q} = \mathbf{Se}_i \mathbf{U} diag(\mathbf{U}^T \mathbf{x}(0)) \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^h \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^h \\ & \cdots\cdots\cdots & & & \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^h \end{pmatrix}$$

évidemment, $rank(\mathbf{Se}_i \mathbf{U}) = d_i + 1$. Comme montré dans (B.3.15), $\mathbf{Q}_i$ devrait être une matrice de rang rangée plein pour assurer la contrainte $\mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}$ pour être satisfaits. Conséquent, $\mathbf{Q}_i = \begin{bmatrix} \mathbf{q}_i \\ \mathbf{q}_j \end{bmatrix}$, $j \in N_i$ peut être établie en sélectionnant le $\mathbf{q}_j, j \in N_i$ jusqu'à ce que le $rank(\mathbf{Q}_i)$ ne peut plus augmenter.

Il y a beaucoup de règles qui peuvent être appliquées pour le choix de $\mathbf{q}_j$. Cependant, dans notre étude, nous choisissons $\mathbf{q}_j$ basé sur le conditionnement établi $\mathbf{Q}_i$. Ce qui signifie que, $\mathbf{q}_j$ est choisi de manière que le conditionnement établi $\mathbf{Q}_i$ est le plus petit.

De (B.3.5), sachant que les coefficients c_k ont un motif de signe alternatif, le problème peut être reformulé comme suit:

$$\min_{\mathbf{c}_i \in I\!R^{(h+1)}, i=1,\ldots,N} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in N_i} \|\mathbf{c}_i - \mathbf{c}_j\|^2 . \tag{B.3.16}$$

$$\text{subject to} \quad \mathbf{c}_i \in C_i = \{\mathbf{c}_i \in I\!R^{(h+1)} : \mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}\}, \ i = 1, 2, \cdots, N \tag{B.3.17}$$

$$c_{ik} = \begin{cases} -c_{ik} & \text{if } k \text{ is odd,} \\ c_{ik} & \text{if } k \text{ is even} \end{cases} \tag{B.3.18}$$

Le problème ici est d'optimiser la somme des fonctions objectifs convexes correspondant aux nœuds dans le réseau. Par contraste avec la méthode directe précédente, le prob-

lème d'optimisation (B.3.16) est convexe. Pour le résoudre, nous allons recourir à une méthode de sous-gradient projeté inspiré de [57] et une méthode de direction alternée de multiplicateurs [11, 22].

(A): ***Méthode de sous-gradient distribuée***

L'idée principale de la méthode de sous-gradient projeté distribuée est de permettre à chaque nœud i de mettre à jour son estimation après deux étapes.

Tout d'abord, une étape de sous-gradient est prise en minimisant la fonction objective locale $\frac{1}{2}\sum_{j\in N_i}\|\mathbf{c}_i-\mathbf{c}_j\|^2$. Ensuite, le résultat intermédiaire est projeté sur l'ensemble des contraintes C_i. Par conséquent, la première étape déduit:

$$\hat{\mathbf{c}}_i[t+1]=\mathbf{c}_i[t]-\gamma[t]\sum_{j\in N_i}\left(\mathbf{c}_i[t]-\mathbf{c}_j[t]\right), \tag{B.3.19}$$

où $\gamma[t]>0$ est un stepsize, tandis que la deuxième étape nécessite le calcul de la projection $\mathbf{c_i}[t+1]=\Omega_{C_i}[\hat{\mathbf{c}}_i[t+1]]$.Cette étape de projection est équivalente à résoudre un problème d'optimisation sous contrainte formulée comme:

$$\min \quad \frac{1}{2}\|\mathbf{c}_i[t+1]-\hat{\mathbf{c}}_i[t+1]\|^2 \text{ s.t.} \quad \mathbf{Q}_i\mathbf{c}_i[t+1]=\bar{x}\mathbf{1}.$$

Pour résoudre ce problème d'optimisation sous contrainte, de manière générale, la fonction Lagrangien augmenté est défini comme suit:

$$L(\mathbf{c}_i[t+1],\mathbf{y})=\frac{1}{2}\|\mathbf{c}_i[t+1]-\hat{\mathbf{c}}_i[t+1]\|^2+\mathbf{y}^T(\mathbf{Q}_i\mathbf{c}_i[t+1]-\bar{x}\mathbf{1}).$$

Chaque nœud met à jour son vecteur de paramètres en utilisant:

$$\mathbf{c}_i[t+1]=\tilde{\mathbf{Q}}_i\bar{\mathbf{x}}_i+(\mathbf{I}_{h+1}-\tilde{\mathbf{Q}}_i\mathbf{Q}_i)\hat{\mathbf{c}}_i[t+1], \tag{B.3.20}$$

avec $\tilde{\mathbf{Q}}_i=\mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$, à condition que $h\geq d_i$ et $\mathbf{Q}_i$ rang rangée plein. Ensuite, nous projetons $\mathbf{c}_i[t+1]$ ur le signe contrainte comme:

$$c_{i,k}[t+1]=\begin{cases}\max(0,c_{i,k}[t+1]) & \text{if } mod(k,2)=0.\\ \min(0,c_{i,k}[t+1]) & \text{if } mod(k,2)\neq 0.\end{cases} \tag{B.3.21}$$

De [57], l'analyse de la convergence de cette méthode a été bien étudié. Avec un choix approprié du stepsize $\gamma[0]$, l'algorithme 5 converge vers un minimum global. La vitesse de convergence dépend encore le choix du stepsize gradient.

(B):***Méthode de la direction alternée des multiplicateurs (ADMM).***

ADMM est relativement facile à mettre en œuvre et ses propriétés de convergence ont été bien étudiée dans des travaux récents tels que, dans [8, 11, 23]. Les avantages de

cette méthode est qu'elle garantie de converger pour tous les paramètres [11] tandis que d'autres techniques telles que la projection de sous-gradient ou double décomposition dépend du choix du stepsize pour les mises à jour de gradient. Contrairement à d'autres méthodes, ADMM a seulement d'un seul paramètre de pénalité ρ qui peut être influencée sur la vitesse de convergence. ADMM assure une très bonne vitesse de convergence lorsque ses paramètres sont choisis de manière appropriée. Il existe quelques travaux traitant de la sélection des paramètres de pénalité pour accélérer la vitesse de ADMM [8, 31, 71]. L'analyse de convergence est également étudié dans [8, 11, 23]. Cependant, dans cette étude, notre paramètre de pénalité est une constante fixe.

L'algorithme de ADMM actes en introduisant certaines variables auxiliaires $\mathbf{z}_{ij}$, de sorte qu'un problème d'optimisation strictement équivalente peut être formulé comme suit:

$$\begin{aligned} \min_{\mathbf{c}_i \in I\!R^{(h+1)}, i=1,\dots,N} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i - \mathbf{c}_j\|^2 . \\ \text{subject to} \quad & \mathbf{c}_i = \mathbf{z}_{ij},\ j \in N_i, \\ & \mathbf{z}_{ji} = \mathbf{z}_{ij}, \\ & \mathbf{z}_{ij} = sign(k) z_{ij}^k, \\ & \mathbf{c}_i \in C_i = \{\mathbf{c} \in I\!R^{(h+1)} : \mathbf{Q}_i \mathbf{c}_i = \bar{x}\mathbf{1}\},\ \ i = 1, 2, \cdots, N. \end{aligned} \tag{B.3.22}$$

où $sign(k) = \begin{cases} 1 & \text{if } k \text{ is even}, \\ -1 & \text{if } k \text{ is odd} \end{cases}$.

La fonction Lagrangien augmenté correspondante est donnée par:

$$L_\rho(\mathbf{c}, \mathbf{z}, \mathbf{y}) = \sum_{i=1}^{N} L_\rho^{(i)}(\mathbf{c}_i, \mathbf{z}, \mathbf{y}),$$

avec

$$L_\rho^{(i)}(\mathbf{c}_i, \mathbf{z}, \mathbf{y}) = \sum_{j\in N_i}\left(\frac{1}{2}\|\mathbf{c}_i - \mathbf{c}_j\|^2 + \mathbf{y}_{ij}^T(\mathbf{c}_i - \mathbf{z}_{ij}) + \frac{\rho}{2}\|\mathbf{c}_i - \mathbf{z}_{ij}\|^2\right), \tag{B.3.23}$$

avec ρ-un coefficient de pénalité. Contrairement à la méthode des multiplicateurs de Lagrange, dans ADMM la minimisation de la fonction de coût selon les variables primaires $\mathbf{c}_i$ et $\mathbf{z}_{ij}$ est effectuée de façon séquentielle. Donc, étant donné les variables primales et les variables duales $\mathbf{y}_{ij}$, la fonction de coût est d'abord minimisé selon à $\mathbf{c}_i$ et la solution obtenue projetée sur l'ensemble des contraintes (c-minimisation). Ensuite $\mathbf{z}_{ij}$ sont obtenus à partir des variables duales données et les valeurs mises à jour de $\mathbf{c}_i$ (z-minimisation). La troisième étape correspond à la mise à jour des variables duales de celles primaires mises à jour (y-minimisation):

$$\mathbf{c}_i[t+1] \quad = \Omega_{C_i}[\text{argmin} \quad L_\rho(\hat{\mathbf{c}}_i[t+1], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t])]; \tag{B.3.24}$$

$$\mathbf{z}_{ij}[t+1] = \text{argmin} \quad L_\rho(\mathbf{c}_i[t+1], \mathbf{z}_{ij}, \mathbf{y}_{ij}[t])]; \tag{B.3.25}$$
$$\mathbf{y}_{ij}[t+1] = \text{argmin} \quad L_\rho(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_{ij}[t])]. \tag{B.3.26}$$

Ces itérations indiquent que la méthode est particulièrement utile lorsque les $c-$ et $z-$ minimisations sont réalisée de manière efficace.

Pour la fonction de coût considéré ici, nous pouvons noter que le problème de c-minimisation (B.3.24) est résolu par:

$$\hat{\mathbf{c}}_i[t+1] = (d_i(1+\rho))^{-1}(\sum_{j\in N_i} \mathbf{c}_j[t] + \rho \mathbf{z}_{ij}[t] - \sum_{j\in N_i} \mathbf{y}_{ij}[t]), \tag{B.3.27}$$

d_i étant le degré de nœud i. Puis, comme pour le cas du sous-gradient projeté nous obtenons la même expression pour $\mathbf{c}_i[t+1]$ comme dans (B.3.20).

Ensuite, nous résolvons le sous-optimisation (B.3.25) analogiquement, ce qui signifie que $\nabla_{\mathbf{z}_{ij}} L(\mathbf{c}_i[k+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_i[t]) = 0$ en ce qui concerne le terme que $\mathbf{z}_{ij} = \mathbf{z}_{ji}$. Réécrire la fonction Lagrange augmentée:

$$L_\rho(\mathbf{c}_i, \mathbf{z}, \mathbf{y}) = \sum_{j\in N_i} \left(\frac{1}{2} \|\mathbf{c}_i - \mathbf{c}_j\|^2 + \mathbf{y}_{ij}^T (\mathbf{c}_i - \mathbf{z}_{ij}) + \frac{\rho}{2} \|\mathbf{c}_i - \mathbf{z}_{ij}\|^2 + \mathbf{y}_{ji}^T (\mathbf{c}_j - \mathbf{z}_{ji}) + \frac{\rho}{2} \|\mathbf{c}_j - \mathbf{z}_{ji}\|^2 \right),$$

Puisque $\nabla_{\mathbf{z}_{ij}} L(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t+1], \mathbf{y}_i[t]) = 0$ avec $\mathbf{z}_{ij} = \mathbf{z}_{ji}$, il est équivalent à:

$$\begin{aligned} & -(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]) - \rho(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + 2\rho \mathbf{z}_{ij}[t+1] = 0 \\ \Leftrightarrow \quad & \mathbf{z}_{ij}[t+1] = \tfrac{1}{2}(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + \tfrac{1}{2\rho}(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]). \end{aligned} \tag{B.3.28}$$

Ensuite, nous projetons $\mathbf{z}_{ij}[t+1]$ sur la contrainte de signe comme:

$$z_{ij}^k[t+1] = \begin{cases} \max(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) = 0. \\ \min(0, z_{ij}^k[t+1]) & \text{if } mod(k,2) \neq 0. \end{cases} \tag{B.3.29}$$

Multiplicateurs de Lagrange sont mis à jour:

$$\mathbf{y}_{ij}[t+1] = \mathbf{y}_{ij}[t] + \rho(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]). \tag{B.3.30}$$

L'algorithme correspondante est alors décrit comme dans l'algorithm 6.

B.3.2.4 La factorisation de la matrice moyenne sur la base de Laplace avec une connaissance imparfaite de $\bar{x}$

Pour toutes les méthodes proposées dans le cas précédente, la valeur de consensus $\bar{x}$ est préalablement un paramètre obligatoire.

Nous détendons maintenant cette exigence au moyen d'une nouvelle méthode qui traite l'estimation des valeurs propres du laplacien quand une valeur de consensus de moyenne exacte n'est pas connue.

Dans ce qui suit, nous supposons que les nœuds exécutent un protocole standard de consensus de moyenne (1.2.1), qui est arrêté après M itérations, et rangent leurs valeurs locales $x_{i,M}$. Dans cette étude, nous voulons montrer la performance de l'algorithme proposé en ce qui concerne certaines valeurs de M.

Puisque on ne sait pas exactement $\bar{\mathbf{x}}$, pour une valeur suffisamment grande de M, $\mathbf{x}(M)$ peut être considéré comme son approximation rationnelle. Conséquent, notre objectif est de trouver les coefficients c_k qui minimisent l'erreur quadratique sur la valeur finale de consensus:

$$E(\mathbf{c}) = \left\| \sum_{k=0}^{h} c_k \mathbf{q}(k) - \mathbf{x}(M) \right\|^2 . \tag{B.3.31}$$

Ensuite, nous pouvons en déduire l'ensemble $\mathbf{S}_1 = \{\boldsymbol{\alpha}_k\}_{k=1,\cdots,h}$ et celle de l'inverse des valeurs propres du laplacien $\mathbf{S}_2 = \frac{1}{\lambda_i}, i = 2, \ldots, D$, qui appartient à $\mathbf{S}_1 : \mathbf{S}_2 \subset \mathbf{S}_1$.

Afin de résoudre le problème d'une manière répartie, le problème d'optimisation peut être reformulé comme suit:

$$\begin{aligned} \min_{\boldsymbol{c}_k \in I\!R^{N\times 1}, k=0,\ldots,h} \quad & \frac{1}{2}\sum_{i=1}^{N}(\sum_{k=0}^{h} c_{i,k} q_{i,k} - \bar{x}_{i,M})^2, \\ \text{subject to} \quad & c_{i,k} = c_{j,k}, \quad i = 1, \ldots, N; j \in N_i; k = 0, \ldots, h. \\ & \mathbf{c}_i = (c_{i,0}, c_{i,1}, \cdots, c_{i,h})^T \in C_i, \end{aligned} \tag{B.3.32}$$

où $c_{i,k}$ le k^{th} coefficient de polynôme local et C_i signifie l'ensemble des contraintes en ce qui concerne nœud i. On le définit comme $C_i = \{\mathbf{c}_i \in I\!R^{h+1}, i = 1, \ldots, N : \mathbf{q}_l^T \mathbf{c}_i = \bar{x}_{i,M}$ with $\mathbf{q}_l \in I\!R^{h+1},\ l \in N_i \cup i\}$ ou de façon équivalente comme $C_i = \{\mathbf{c}_i \in I\!R^{h+1}, i = 1, \ldots, N : \mathbf{Q}_i \mathbf{c}_i = \bar{x}_{i,M}\mathbf{1},\ \mathbf{Q}_i \in I\!R^{d_i+1\times h+1}\}$ où $\mathbf{Q}_i = [\mathbf{q}(0)\ \mathbf{q}(1) \ldots \mathbf{q}(h)]$, étant rang plein de rangée en construction.

Cet ensemble de contraintes permet de respecter non seulement l'égalité du vecteur des coefficients locaux $\mathbf{c}_i$ si dans un voisinage donné, mais aussi pour faire appliquer le produit scalaire $\mathbf{q}_i^T \mathbf{c}_i$ être égal globalement. a noter que, en introduisant les variables auxiliaires $\{\mathbf{z}_{ij}\}$, le problème d'optimisation convexe sous contrainte suivant est absolument équivalent au problème (B.3.32):

$$\min_{\boldsymbol{c}_k \in I\!R^{N\times 1}, k=0,\ldots,h} \quad \frac{1}{2}\sum_{i=1}^{N}(\sum_{r=0}^{h} c_{i,k} q_{i,k} - \bar{x}_{i,M})^2. \tag{B.3.33}$$

$$\begin{aligned} \text{subject to} \quad & c_{i,k} = z_{ij}^k, \quad i = 1, \ldots, N; j \in N_i \\ & z_{ji}^k = z_{ij}^k, \quad k = 0, \ldots, h. \\ & \mathbf{z}_{ij}^k = z_{ij}^k, \quad \text{if } k \text{ is even}, \end{aligned} \tag{B.3.34}$$

$$\begin{aligned}&\mathbf{z}_{ij}^k = -z_{ij}^k, \quad \text{if } k \text{ is odd},\\ &\mathbf{c}_i \in C_i.\end{aligned}$$

Puisque le graphe est supposé être connecté, les contraintes (B.3.34) forcent $\mathbf{c}_i = \mathbf{c}_j, j \in N_i$. Nous pouvons alors écrire le Lagrangien augmenté associé:

$$L_\rho(\mathbf{c},\mathbf{z},\mathbf{y}) = \sum_{i=1}^{N}\left[\frac{1}{2}\sum_{k=0}^{h}(c_{i,k}q_{i,k}-\bar{x}_{i,M})^2+\sum_{k=0}^{h}\left(\sum_{j\in N_i}y_{ij}^k(c_{i,k}-z_{ij,k})\right)+\sum_{j\in N_i}\frac{\rho}{2}\|\mathbf{c}_i-\mathbf{z}_{ij}\|^2\right].$$

ou sous forme de vecteur:

$$L_\rho(\mathbf{c},\mathbf{z},\mathbf{y}) = \sum_{i=1}^{N}\left[\frac{1}{2}(\mathbf{q}_i^T\mathbf{c}_i-\bar{x}_{i,M})^2+\sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i-\mathbf{z}_{ij})+\sum_{j\in N_i}\frac{\rho}{2}\|\mathbf{c}_i-\mathbf{z}_{ij}\|^2\right].$$

La solution à ce problème ADMM agit en trois étapes:

- Tout d'abord, la minimisation de $\mathbf{c}_i$:

$$\mathbf{c}_i[t+1] = \Omega_{C_i}[\text{argmin} \quad L(\mathbf{c}_i,\mathbf{z}_{ij}[t],\mathbf{y}_i[t])]. \tag{B.3.35}$$

  où $\Omega_{C_i}[\cdot]$ représente la projection dans l'ensemble des contraintes du vecteur dans l'argumentation.

- Deuxièmement, la minimisation en ce qui concerne à $\mathbf{z}_{ji} = \mathbf{z}_{ij}$:

$$\mathbf{z}_{ij}[t+1] = \text{argmin} \quad L(\mathbf{c}_i[t+1],\mathbf{z}_{ij},\mathbf{y}_i[t]). \tag{B.3.36}$$

- Troisième, la mise à jour de multiplicateur de Lagrange:

$$\mathbf{y}_{ij}[t+1] = \mathbf{y}_{ij}[t]+\rho(\mathbf{c}_i[t+1]-\mathbf{z}_{ij}[t+1]). \tag{B.3.37}$$

Résoudre le problème sous-optimisation (B.3.35) par $\frac{\partial L_\rho(\mathbf{c},\mathbf{z},\mathbf{y})}{\partial \mathbf{c}_i} = 0$, nous obtenons:

$$(\mathbf{q}_i\mathbf{q}_i^T-\bar{x}_{i,M})\mathbf{q}_i+\sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i-\mathbf{z}_{ij})+\frac{\rho}{2}\sum_{j\in N_i}(\mathbf{c}_i-\mathbf{z}_{ij})^2 = 0.$$

$$\Leftrightarrow (\mathbf{q}_i\mathbf{q}_i^T+\rho d_i\mathbf{I}_{h+1})\mathbf{c}_i = \bar{x}_{i,M}\mathbf{q}_i+\rho\sum_{j\in N_i}\mathbf{z}_{ij}-\sum_{j\in N_i}\mathbf{y}_{ij}.$$

Puis,

$$\hat{\mathbf{c}}_i[t+1] = (\mathbf{q}_i\mathbf{q}_i^T+\rho d_i\mathbf{I}_{h+1})^{-1}(\bar{x}_{i,M}\mathbf{q}_i+\rho\sum_{j\in N_i}\mathbf{z}_{ij}[t]-\sum_{j\in N_i}\mathbf{y}_{ij}[t]).$$

Puis, projeter la solution obtenue à l'ensemble des contraintes

$$\mathbf{c}_i[t+1] = \Omega_{C_i}[\hat{\mathbf{c}}_i[t+1]],$$

nous obtenons:

$$\mathbf{c}_i[t+1] = \tilde{\mathbf{Q}}_i \bar{\mathbf{x}}_{i,M} + (\mathbf{I}_{h+1} - \tilde{\mathbf{Q}}_i \mathbf{Q})\hat{\mathbf{c}}_i[t+1], \tag{B.3.38}$$

avec $\tilde{\mathbf{Q}}_i = \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$ et $\mathbf{Q}_i$ être une matrice de rang plein rangée.

Ensuite, résoudre le problème sous-optimisation (B.3.36) déduit

$$\mathbf{z}_{ij}[t+1] = \frac{1}{2}(\mathbf{c}_i[t+1] + \mathbf{c}_j[t+1]) + \frac{1}{2\rho}(\mathbf{y}_{ij}[t] + \mathbf{y}_{ji}[t]). \tag{B.3.39}$$

L'algorithme correspondante est alors décrit comme dans l'algorithm 7. Quand les coefficients c_k sont obtenus approximativement, nous pouvons utiliser des Algorithmes 9 et 10 pour estimer tout le spectre de matrice de Laplace $\mathbf{L}$, qui peut être utilisé pour estimer les indices de la robustesse du réseau.

B.3.2.5 La factorisation de la matrice moyenne sur la base de Laplace avec une valeur de consensus inconnue $\bar{x}$

Selon la section précédente, une façon pour l'estimation des valeurs propres du laplacien est:

- résoudre le problème du consensus de moyenne.
- résoudre le problème de la factorisation de la matrice de moyennage (B.3.15), Algorithmes 5 et 6.
- récupérer les valeurs propres du laplacien en utilisant les algorithmes 9 et 10.

Au lieu de suivre ces étapes, dans cette section, nous proposons une méthode pour fusionner les deux premières étapes. Dans ce but, nous proposons de résoudre une combinaison convexe des deux fonctions convexes, c'est-à-dire

$$\begin{aligned} \min_{\bar{x}_i, \mathbf{c}_i, i=1,\cdots,N} \quad & \frac{\theta}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i - \mathbf{c}_j\|^2 + \frac{1-\theta}{2}\sum_{i=1}^{N}(\bar{x}_i - x_i(0))^2, \\ \text{s.t.} \quad & \mathbf{Q}_i\mathbf{c}_i = \bar{x}_i\mathbf{1}, \quad i = 1,\ldots,N, \end{aligned} \tag{B.3.40}$$

avec θ-un paramètre scalaire ajustable ($0 \leq \theta \leq 1$), $\bar{x}_i$-une estimation locale de la valeur moyenne, tandis que $x_i(0)$ est une valeur initiale fixe de nœud i.

On voit que, la fonction d'objectif (B.3.40) est une combinaison de deux fonctions convexes. La première prend en compte le problème du consensus de moyenne, tandis que la seconde concerne l'estimation des coefficients du polynôme dont les racines sont

l'inverse des valeurs propres du laplacien distinctes différents de zéro. Et ce problème d'optimisation convexe peut être résolu efficacement en utilisant l'approche de l'ADMM.

Afin de dériver un algorithme ADMM pour le problème (B.3.40), nous introduisons les variables auxiliaires $\{\mathbf{z}_{ij}, \mu_{ij}\}$. Par conséquent, le problème d'optimisation convexe sous contrainte suivant est absolument équivalente au problème (B.3.40):

$$\begin{aligned} \min_{\bar{x}_i, \mathbf{c}_i, i=1,\cdots,N} \quad & \frac{\theta}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i-\mathbf{c}_j\|^2+\frac{1-\theta}{2}\sum_{i=1}^{N}(\bar{x}_i-x_i(0))^2, \\ \text{s.t} \quad & \bar{x}_i=\mu_{ij}, \quad i=1,\ldots,N, \quad j\in N_i. \\ & \mathbf{c}_i=\mathbf{z}_{ij}. \\ & \mathbf{z}_{ij}=sign(k)z_{ij}^k, \\ & \mathbf{c}_i\in C_i(\bar{x}_i). \end{aligned}$$

où

- $sign(k)=\begin{cases} 1 & \text{if } k \text{ is even}, \\ -1 & \text{if } k \text{ is odd} \end{cases}$.
- $C_i(\bar{x}_i)$ signifie l'ensemble des constraintes ec ce qui concern nœud i. On le définit comme $C_i=\{\mathbf{c}\in I\!R^{h+1}|\ \mathbf{Q}_i\mathbf{c}=\bar{x}_i\mathbf{1}\}$.

Maintenant, en introduisant certaines variables duales $\boldsymbol{\upsilon}$ and $\mathbf{y}$, le Lagrangien augmenté est défini comme:

$$L_{\rho_1\rho_2}(\bar{\mathbf{x}},\mathbf{c},\boldsymbol{\mu},\mathbf{z},\boldsymbol{\upsilon},\mathbf{y})=L_{\rho_1}(\mathbf{c},\mathbf{z},\mathbf{y})+L_{\rho_2}(\bar{\mathbf{x}},\boldsymbol{\mu},\boldsymbol{\upsilon}).$$

où:

$$\begin{aligned} L_{\rho_1}(\mathbf{c},\mathbf{z},\mathbf{y}) &= \frac{\theta}{2}\sum_{i=1}^{N}\sum_{j\in N_i}\|\mathbf{c}_i-\mathbf{c}_j\|^2+\sum_{j\in N_i}\frac{\rho_1}{2}\|\mathbf{c}_i-\mathbf{z}_{ij}\|^2+\sum_{j\in N_i}\mathbf{y}_{ij}^T(\mathbf{c}_i-\mathbf{z}_{ij}), \\ L_{\rho_2}(\bar{\mathbf{x}},\boldsymbol{\mu},\boldsymbol{\upsilon}) &= \frac{1-\theta}{2}\sum_{i=1}^{N}(\bar{x}_i-x_i(0))^2+\sum_{j\in N_i}\frac{\rho_2}{2}(\bar{x}_i-\mu_{ij})^2+\sum_{j\in N_i}\upsilon_{ij}(\bar{x}_i-\mu_{ij}), \end{aligned}$$

où ρ_1, ρ_2 sont des termes de pénalité supposés constants.

ADMM fournit une solution itérative qui agit les cinq étapes suivant ci-dessous:

- Minimisation ec ce qui concerm la valeur moyenne du consensus $\bar{x}_i$:

$$\bar{x}_i[t+1]=\arg\min\ L_{\rho_2}(\bar{x}_i[t],\mu_{ij}[t],\upsilon_{ij}[t]). \tag{B.3.41}$$

- Minimisation en ce qui concern les coefficients polynomiaux $\mathbf{c}_i$:

$$\hat{\mathbf{c}}_i = \text{argmin } L_{\rho_1}(\mathbf{c}_i[t], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t]), \tag{B.3.42}$$

$$\mathbf{c}_i[t+1] = \Omega_{C_i(\bar{x}_i[t+1])}[\hat{\mathbf{c}}_i] \tag{B.3.43}$$

  où $\Omega_{C_i(\bar{x}_i[k+1])}[\cdot]$ dresse pour la projection dans l'ensemble des contraintes $C_i(\bar{x}_i[k+1])$ du vecteur dans l'argumentation.

- Minimisation en ce qui concerm les variables auxiliaires μ_{ij} avec la contrainte $\mu_{ji} = \mu_{ij}$:

$$\mu_{ij}[t+1] = \text{argmin } L_{\rho_2}(\bar{x}_i[t+1], \mu_{ij}[t], \upsilon_{ij}[t]). \tag{B.3.44}$$

- Minimisation en ce qui concerm les variables auxiliaires $\mathbf{z}_{ij}$:

$$\mathbf{z}_{ij}[t+1] = \text{argmin } L_{\rho_1}(\mathbf{c}_i[t+1], \mathbf{z}_{ij}[t], \mathbf{y}_{ij}[t]). \tag{B.3.45}$$

- La mise à jour Multiplicateurs de Lagrange:

$$\upsilon_{ij}[t+1] = \upsilon_{ij}[t] + \rho_2(\bar{x}_i[t+1] - \mu_{ij}[t+1]). \tag{B.3.46}$$

$$\mathbf{y}_{ij}[t+1] = \mathbf{y}_{ij}[t] + \rho_1(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]). \tag{B.3.47}$$

Résoudre les problèmes de sous-optimisation (B.3.41) et (B.3.42) est équivalent à résoudre les équations suivantes:

Nous obtenons:

$$\frac{L_{\rho_2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon})}{\partial \bar{x}_i} = 0$$

$$\Leftrightarrow \quad \bar{x}_i[t+1] = \frac{(1-\theta)x_i(0) + \rho_2 \sum_{j\in N_i} \mu_{ij}[t] - \sum_{j\in N_i} \upsilon_{ij}[t]}{1-\theta+\rho_2 d_i}. \tag{B.3.48}$$

Puis, projeter la solution obtenue dans l'ensemble des contraintes:

$$\mathbf{c}_i[k+1] = \Omega_{C_i(\bar{x}_i[k+1])}[\hat{\mathbf{c}}_i[k+1]],$$

nous obtenons:

$$\mathbf{c}_i[k+1] = \tilde{\mathbf{Q}}_i \bar{x}_i[k+1]\mathbf{1} + (\mathbf{I}_{h+1} - \tilde{\mathbf{Q}}_i\mathbf{Q})\hat{\mathbf{c}}_i[k+1], \tag{B.3.49}$$

avec $\tilde{\mathbf{Q}}_i = \mathbf{Q}_i^T(\mathbf{Q}_i\mathbf{Q}_i^T)^{-1}$, $\mathbf{Q}_i$ étant rang plein de rangée en construction.

Ensuite, la solution des problèmes sous-optimisation (B.3.44) et (B.3.45), de la même façon, déduit

1. Le dérivé de $\mathbf{L}_{\rho_2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon})$ en ce qui concern μ_{ij} par rapport à la contrainte $\mu_{ij} = \mu_{ji}$:

$$\frac{\partial L_{\rho_2}(\bar{\mathbf{x}}, \boldsymbol{\mu}, \boldsymbol{\upsilon})}{\partial \mu_{ij}} = -\upsilon_{ij} - \upsilon_{ji} - \rho_2(\bar{x}_i + x_j) + 2\rho_2\mu_{ij} = 0$$

$$\Leftrightarrow \mu_{ij}[t+1] = \frac{\bar{x}_i[t+1] + \bar{x}_j[t+1]}{2} + \frac{\upsilon_{ij}[t] + \upsilon_{ji}[t]}{2\rho_2}. \tag{B.3.50}$$

2. Le dérivé de $\mathbf{L}_{\rho_1}(\mathbf{c}, \mathbf{z}, \mathbf{y})$ en ce qui concern $\mathbf{z}_{ij}$

$$\begin{aligned} & \frac{L_{\rho 1}(\mathbf{c}_i, \mathbf{z}_{ij}, \mathbf{y}_{ij})}{\mathbf{z}_{ij}} = 0 \\ \Leftrightarrow \quad & -\mathbf{y}_{ij}[t] - \rho(\mathbf{c}_i[t+1] - \mathbf{z}_{ij}[t+1]) = 0 \\ \Leftrightarrow \quad & \mathbf{z}_{ij}[t+1] = \mathbf{c}_i[t+1] + \tfrac{1}{\rho_1}\mathbf{y}_{ij}[t]. \end{aligned} \tag{B.3.51}$$

Le paramètre θ permet pondération des deux fonctions combinées de coûts. Puisque l'ensemble des contraintes $C_i(\bar{x}_i)$ dépend de la valeur de consensus $\bar{x}_i$. La procédure d'optimisation peut d'abord donner plus de poids au problème de consensus de moyenne, puis au problème de la factorisation. Dans ce but, au lieu de garder θ constant, nous adoptons un temps coefficient variant pris comme

$$\theta(t) = \frac{1 - e^{-\beta t}}{1 + e^{-\beta t}} \quad (0 < \beta < 1) \tag{B.3.52}$$

pour le garder dans la gamme $[0, 1]$ et d'augmenter l'importance de la factorisation progressivement tout en diminuant celle de le consensus de moyenne.

Remark 16

Pratiquement, nous pouvons voir que le terme de consensus de moyenne converge beaucoup plus rapide que le problème de la factorisation de la matrice moyenne. Ainsi, θ peut être choisi près de 1.

L'algorithme correspondante est alors décrit comme dans l'algorithm 8.

B.3.2.6 Récupération du spectre de Laplace

A chaque nœud, l'ensemble $\mathbf{S}_1$ du step-size obtenu au moyen de les algorithmes proposées (Algorithmes 4, 5, 6, 7, 8) contient l'ensemble $\mathbf{S}_2$ de l'inverse des valeurs propres du laplacien distinctes différents de zéro.

Soit $\hat{\bar{x}}_i$être la valeur finale de consensus reconstruit par $\hat{\bar{x}}_i = \sum_{k=0}^{h} c_k q_{i,k} = \mathcal{P}_i(\mathbf{c}, \mathbf{q}_i)$ avec les coefficients obtenus au moyen d'algorithmes 4, 5, 6, 7 et 8.

Suite à la Proposition 8, l'idée est de réduire, étape par étape, le degré du polynôme $\mathcal{P}_i(\mathbf{c}, \mathbf{q}_i)$ par enlever un élément de $\mathbf{S}_1$ à chaque étape. Nous savons que si l'élément

enlevé est également un élément de $\mathbf{S}_2$ puis $\hat{\tilde{x}}_i \neq \tilde{\mathcal{P}}_i(\mathbf{c}, \tilde{\mathbf{q}}_i)$ où $\tilde{\mathcal{P}}_i$ est le polynôme de degré réduit en ce qui concern nœud i. Algorithme 9 décrit la procédure proposée.

Ensuite, nous obtenons l'ensemble des valeurs propres du laplacien distincts différents de zéro $\Lambda = \{\lambda_2, \ldots, \lambda_{D+1}\}$. Afin d'obtenir tout le spectre $Sp(\mathbf{L}) = \{\lambda_1^{m_1}, \lambda_2^{m_2}, \ldots, \lambda_{D+1}^{m_{D+1}}\}$, les multiplicités m_i devraient être définis. Puisque, elles sont toutes entier, le problème a recours actuellement à un problème de la programmation en nombres entiers.

Étant donné l'ensemble des valeurs propres du laplacien distinctes différents de zéro Λ, d_i le degré de chaque nœud i, inspiré d'une propriété intéressante $\sum_{i=1}^{N} \lambda_i(\mathbf{L}) = \sum_{i=1}^{N} d_i$ (voir dans Section 2.3), l'optimisation est maintenant décrit comme suit:

$$\begin{aligned} \min_{\mathbf{m}\in\mathbb{R}^D} \quad & \Lambda^T\mathbf{m} - \sum_{i=1}^{N} d_i, \\ \text{subject to} \quad & \sum_{j=1}^{D} m_j = N-1,\ j = 1, \ldots, D. \\ & m_j \geq 1, \text{integer}. \end{aligned} \tag{B.3.53}$$

où $D = |\Lambda|$ et $\mathbf{m}$ est le vecteur des multiplicités des valeurs propres.

Dans la littérature, la méthode de séparation et d'évaluation progressive est la technique basique pour résoudre le problème de programmation entier. La programmation linéaire (LP) de la relaxation (sans restriction de nombre entier) est adoptée pour estimer la solution optimale d'une programmation en nombres entiers [13]. Conséquent, le problème (B.3.53) est maintenant réécrit strictement équivalente sous la forme de programmation linéaire comme suit:

$$\begin{aligned} \min_{\mathbf{m}\in\mathbf{R}^D} \quad & \Lambda^T\mathbf{m}, \\ \text{subject to} \quad & \sum_{j=1}^{D} m_j = N-1; \\ & \Lambda^T\mathbf{m} = \sum_{i=1}^{N} d_i, \qquad (\mathbf{M}) \\ & m_j \geq 1,\ \text{integer}. \end{aligned} \tag{B.3.54}$$

Où $\mathbf{M}$ est l'ensemble des contraintes sans restriction en nombres entiers. De ce fait, $\mathbf{M}$ peut être exprimée comme $\mathbf{M} = \{\mathbf{m} \in \mathbb{R}^D | \sum_{j=1}^{D} m_j = N-1, \Lambda^T\mathbf{m} = \sum_{i=1}^{N} d_i \text{ and } m \geq 1\}$. Le problème (B.3.54) sans restriction en nombres entiers est appelé la programmation linéaire (LP) de la relaxation.

La Algorithme de séparation et d'évaluation progressive est décrit comme dans l'algorithme 10.

B.4 Reconstruction de la topologie du réseaux

B.4.1 Introduction

Une topologie de réseau est en informatique une définition de l'architecture (physique ou logique) d'un réseau, définissant les connexions entre ces postes et une hiérarchie éventuelle entre eux ; elle peut définir la façon dont les équipements sont interconnectés ou la représentation, spatial du réseau (topologie physique) ou la façon dont les données transitent dans les lignes de communication (topologie logique).

L'identification de la topologie de réseau se réfère à détecter et identifier les éléments de réseau intéressés et la relation entre les éléments de réseau cible et représente la construction de la topologie sous une forme appropriée. Par conséquent, l'identification des réseaux de systèmes devient une tâche de plus en plus attrayant pour résoudre de nombreux problèmes dans différents domaines scientifiques et techniques, par exemple, en biologie (biochimique, de neurones et des réseaux écologiques [2]), la finance, l'informatique (Internet et le World Wide Web), le transport (livraison et réseau de distribution), et de l'ingénierie électrique.

Par exemple, l'architecture d'un réseau de recouvrement- comment il alloue les adresses, etc.- peut être optimisé significativement par la connaissance de la distribution et la connectivité des nœuds sur le réseau de sous-couche qui transporte effectivement le trafic. Plusieurs systèmes importants, tels que l'initiative P4P et RMTP, utilisent des informations sur la topologie du réseau sous-couche pour l'optimisation ainsi que la gestion [1].

Dans l'Internet, par exemple, le mécanisme habituel pour générer la topologie d'un réseau par l'utilisation d'Traceroute. Traceroute est exécuté sur un noeud, appelé la source, en spécifiant l'adresse de destination d'un noeud. Cette exécution produit une séquence d'identifiants, appelé trace, correspondant à la route empruntée par les paquets qui voyagent à partir de la source vers la destination. Un ensemble de trace T est générée en exécutant plusieurs reprises traceroute sur un réseau, en variant les noeuds terminaux, ce est à dire la source et la destination. Si T contient des traces qui identifient chaque cas lorsqu'un bord est incident sur un nœud, il est possible de reconstruire le réseau exactement [1].

Suite à notre étude, à partir d'un réseau de consensus général et les mesures de consensus, nous traitons le problème de déduire la topologie correcte du réseau en présence de noeuds anonymes (ou noeuds non marqués). En d'autres termes, est-il possible pour un noeud arbitraire j de reconstruire le réseau correct par des mesures consensuelles moyennes?

B.5 La solution distribuée de reconstruction de la topologie du réseau

Dans le chapitre précédent, nous savons maintenant que noeud i peux obtenir le spectre du laplacien par des mesures consensuelles moyennes. Nous affirmons l'hypothèse suivante:

Hypothèse 1: Toutes les valeurs propres non nulles de Laplace sont distincts.

En considérant un réseau représenté avec un graphe $G(V, E)$, nous rappelons que à l'instant de temps k,

$$\mathbf{x}(k) = \mathbf{W}\mathbf{x}(k-1)$$

ou de façon équivalente

$$\mathbf{x}(k) = \mathbf{W}^k\mathbf{x}(0)$$

Considérons la décomposition suivante valeur propre de la matrice de consensus $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{U}^T. \tag{B.5.1}$$

avec $\mathbf{D} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$ et $\mathbf{\Delta}$ est la matrice diagonale des valeurs propres de $\mathbf{L}$. ϵ peut être choisi comme $\frac{2}{\lambda_2(\mathbf{L})+\lambda_N(\mathbf{L})}$) [74], puis $\lambda(\mathbf{W}) = 1 - \epsilon\lambda(\mathbf{L})$.

Étant donné que les vecteurs propres laplaciennes forment une base de $\mathbb{R}^N$, la condition initiale $\mathbf{x}(0)$ peut être étendu telle que $\mathbf{x}(0) = \sum_{i=1}^{N} \beta_i \mathbf{U}_{.i} = \mathbf{U}\mathbf{b}$, où $\mathbf{b} = \begin{pmatrix} \beta_1 & \beta_2 & \cdots & \beta_N \end{pmatrix}^T$ contient les coefficients de expansion, et $\mathbf{U}_{.i}$ est le i-ere vecteur propre, i.e. le i-ere colonne de $\mathbf{U}$.

Par conséquent:

$$\mathbf{x}(k) = \mathbf{U}\mathbf{D}^k\mathbf{U}^T\mathbf{U}\mathbf{b} = \mathbf{U}\mathbf{D}^k\mathbf{b} = \mathbf{U}diag(\mathbf{b})vecd(\mathbf{D}^k),$$

où $vecd(\cdot)$ est le vecteur colonne construit avec les éléments diagonaux de la matrice en argument.

Nous pouvons écrire l'équation équivalente ci-dessus:

$$\mathbf{x}(k) = \tilde{\mathbf{U}}vecd(\mathbf{D}^k), \quad \text{avec} \quad \tilde{\mathbf{U}} = \mathbf{U}diag(\mathbf{b})$$

Par conséquent, $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T = \mathbf{U}diag(\mathbf{b}^2)\mathbf{U}^T$ et $\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} = diag(\mathbf{b}^2)$. Nous pouvons conclure que:

$$\mathbf{W} = \tilde{\mathbf{U}}\mathbf{D}diag(\mathbf{b}^2)^{-1}\tilde{\mathbf{U}}^T. \tag{B.5.2}$$

B.5.1 Estimation des vecteurs propres de la matrice de consensus basée matrice de Laplace

Au noeud j, l'état à l'instant k est alors donnée par:

$$x_j(k) = \sum_{i=1}^{N} \lambda_i^k(\mathbf{W})\beta_i u_{j,i},$$

En empilant N mesures consécutives noeud j obtient:

$$\begin{pmatrix} x_j(0) \\ x_j(1) \\ \vdots \\ x_j(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \lambda_2(\mathbf{W}) & \cdots & \lambda_N(\mathbf{W}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_2^{N-1}(\mathbf{W}) & \cdots & \lambda_N^{N-1}(\mathbf{W}) \end{pmatrix} \begin{pmatrix} \tilde{u}_{j,1} \\ \tilde{u}_{j,2} \\ \vdots \\ \tilde{u}_{j,N} \end{pmatrix}, \quad \tilde{u}_{j,i} = u_{j,i}\beta_i$$

ou de façon équivalente

$$\mathbf{x}_j = \boldsymbol{\Upsilon}\tilde{\mathbf{U}}_{j\cdot}^T \tag{B.5.3}$$

De Hypothèse 1, nous pouvons conclure que $\boldsymbol{\Upsilon}$ est une matrice de rang plein. Par conséquent, noeud j peut résoudre le problème ci-dessus efficacement et obtenir:

$$\tilde{\mathbf{U}}_{j\cdot}^T = \boldsymbol{\Upsilon}^{-1}\mathbf{x}_j. \tag{B.5.4}$$

Maintenant, le nœud ne connaît que le j-ere rangée de $\tilde{\mathbf{U}}$.

Considérons un message passant schéma où les nœuds envoient à leurs voisins les rangées qu'ils ont.

Puis, un certain nombre d'échanges des messages, au moins égale au diamètre du graphe, noeud j a toutes les rangées de $\tilde{\mathbf{U}}$. Cependant, étant donné que certains des nœuds sont supposés être anonymes, ils ne envoient pas le label de la rangée qu'ils ont. Deux types de messages sont transmis dans le réseau. les rangées associées à des nœuds non-anonymes sont correctement étiquetés tandis que ceux qui sont associés à des nœuds anonymes ne sont pas étiquetés. En conséquence, noeud j ne reçoit $\tilde{\mathbf{U}}$ allant jusqu'à rangées permutation, i.e. il reçoit les $\hat{\mathbf{U}} = \boldsymbol{\Pi}\tilde{\mathbf{U}}$, $\boldsymbol{\Pi}$ étant une matrice de permutation.

B.5.2 Network topology reconstruction

Nous pouvons noter que $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = diag(\mathbf{b}^2)$. Par conséquent, de (B.5.2), nous obtenons:

$$\mathbf{W} = \boldsymbol{\Pi}\hat{\mathbf{W}}\boldsymbol{\Pi}^T, \quad \text{with} \quad \hat{\mathbf{W}} = \hat{\mathbf{U}}\mathbf{D}(\hat{\mathbf{U}}^T\hat{\mathbf{U}})^{-1}\hat{\mathbf{U}}^T. \tag{B.5.5}$$

Nous pouvons conclure que de la matrice estimée $\hat{\mathbf{W}}$, on obtient un graphe qui est isomorphe au graphe original. Dans ce qui suit, nous indiquons deux conditions suffisantes pour assurer la reconstruction correcte du graphique. Nous rappelons d'abord les notions suivantes de observabilité:

Un réseau est dit être [44]:

- noeud observable d'un noeud donné, si ce noeud est en mesure de reconstruire l'état du réseau de ses propres mesures. Cette question a été étudiée par exemple dans [58] and [65] où il a été déclaré que un réseau avec une matrice d'état ayant au moins une valeur propre non simple n'est pas le noeud observable.
- voisinage observable d'un noeud donné si ce nœud peut reconstruire l'état du réseau entier de ses propres mesures et celles de ses voisins. Cette question a été étudiée dans [43].
- globalement observable si elle est la voisinage observable depuis ne importe quel noeud.

Proposition 9

Supposons que le graphe est noeud-observable ou voisinage observable du noeud j. Si toutes les entrées de la condition initiale sont distinctes alors la topologie du réseau peut être reconstruit exactement du noeud j.

Proposition 10

Supposons que le réseau contient un ensemble $\mathcal{A}$ de noeuds anonymes et le graphe est noeud-observable ou voisinage observable du noeud j. Si toutes les entrées de la condition initiale associée aux noeuds anonymes sont toutes distinctes alors la topologie de réseau peut être reconstruit exactement du noeud j.

Ici, l'ambiguïté de permutation est limitée à des entrées associées aux noeuds anonymes.

B.6 Conclusions générales

B.6.1 Résumé des contributions et conclusions

Ce chapitre a pour l'objective de conclure la thèse a travers un bref summerization ainsi qu'une perspective pour des travaux futurs.

Les Problèmes de Consensus en temps fini ont été reçus une vaste attention de la communauté de recherche en raison de ses avantages dans de nombreux domaines d'application, en particulier dans le calcul distribué. Brièvement parlant, les algorithmes

de consensus de moyenne en temps fini (pour lesquelles un consensus peut être obtenu en un nombre fini d'étapes) sont couramment utilisés comme élément de base pour plusieurs algorithmes de contrôle, estimation ou inférence distribués en raison du fait qu'elles peuvent garantir une exécution en temps minimal, ces algorithmes sont donc beaucoup plus attrayant que ceux qui assurent la convergence asymptotique.

Ceci impact sur la thèse avec la conception de protocole de consensus de moyenne en temps fini indépendamment de la matrice de Laplace d'un graphe donné de manière distribuée et son application dans l'évaluation de la robustesse du réseau.

Tout d'abord, nous avons proposé un nouvel algorithme pour le protocole d'auto-configuration d'un consensus de moyenne en temps fini où matrices ne sont pas nécessairement basées sur la matrice de Laplace d'un graphe donné de manière distribuée. Plus précisément, nous avons résolu un problème de la factorisation de la matrice de manière distribuée en utilisant une séquence d'apprentissage. La méthode est basée sur la méthode de rétro-propagation et une méthode de descente de gradient associée. Conceptuellement, chaque itération linéaire $k, k = 0, \ldots, D$ est illustrée comme une couche d'un réseau. L'idée principale est de calculer les dérivées partielles de l'erreur entre la sortie réelle et en sortie souhaité, puis propager les ramener à chaque itération k pour mettre à jour les poids. Étant donné le diamètre du graphe $d(G)$, la valeur optimale du nombre d'étapes nécessaires pour atteindre un consensus de moyenne en faisant varier la valeur de D dans $(d(G) \leq D \leq N-1)$.

Évidemment, il est incommode pour les réseaux de grande envergure. Cependant, un point intéressant peut être indiquée dans ce travail est que les matrices de consensus obtenus ne sont pas des matrices doublement stochastiques. Ceci est la condition de la convergence pour un protocole de consensus selon [74].

Deuxièmement, l'étude sur la robustesse des réseaux de plus en plus attiré un attention de la communauté scientifique. Conséquent, dans cette thèse, l'évaluation de la robustesse du réseau, qui est mis en œuvre par la Résistance effective du graphe ($\mathcal{R}$) et le nombre d'arbres couvrants (ξ) d'une manière distribuée, devient évidemment une application domaine prometteuse. Puisque deux mesures de la robustesse sont des fonctions du spectre de Laplace $sp(\mathbf{L})$, donc la tâche principale est maintenant d'estimer le spectre de Laplace. Précisément, trois cas ont été considérés: parfaitement connue $\bar{x}$, bruyant, et inconnu $\bar{x}$.

La matrice de consensus moyen peut être prise en D la matrice basée sur Laplace de consensus, où D représente le nombre de valeurs propres du laplacien distinctes différents de zéro. Pour le premier cas (parfaitement connue $\bar{x}$), toutes les méthodes sont caractérisées en deux approches, qui sont approche directe et l'approche indirecte. La méthode directe donne lieu à un problème d'optimisation non convexe en ce qui concern les step-sizes $\alpha_k, k = 1, \ldots, N-1$ qui sont l'inverse des valeurs propres du laplacien distinctes

différents de zéro. Puisque, c'est une méthode basée sur gradient, il souffre de convergence lent. Pour améliorer la vitesse de convergence, un problème d'optimisation convexe est introduit pour estimer les coefficients $c_k, k = 0, \ldots, N-1$ d'un polynôme donné dont les racines sont les stepsizes $\alpha_k, k = 1, \ldots, N-1$ qui permet de factoriser la matrice de moyennage $\mathbf{J}_N = \frac{\mathbf{1}\mathbf{1}^T}{N}$. De cette façon, deux méthodes fameux (les méthode de sous-gradient projeté et ADMM) sont appliquées pour résoudre ce problème d'optimisation convexe. Par ces méthodes proposées, les valeurs propres du laplacien distinctes peuvent être déduites. Ensuite, au moyen d'une programmation en nombres entiers, tout le spectre $sp(\mathbf{L})$ est obtenu pour calculer les indices de la robustesse.

L'inconvénient de ces méthodes est l'evolutivité. Pour les graphes de grande échelle, l'estimation de tout le spectre $sp(\mathbf{L})$ est significativement difficile à exécuter. Par conséquent, l'estimation peut être limitée à celle des valeurs propres les plus significatifs pour estimer la limite de l'indice de la robustesse du réseau ($\lambda_2(\mathbf{L})$).

Méthode de sous-gradient projeté est très applicable en raison de sa simplicité. Cependant, l'inconvénient est qu'il peut être lente. Conséquent, il faut faire attention au choix du stepsize $\gamma(t)$. En revanche, l'ADMM est garanti à converger pour tous les paramètres. Il a seulement un seul paramètre de pénalité ρ qui peut être influencée sur la vitesse de convergence. Ainsi, dans les cas restants, ADMM est le meilleur choix.

Dans le cas de la valeur de consensus imparfaite $\bar{x}$,nous avons supposé un protocole standard de consensus a été exécuté et conservé certaines valeurs intermédiaires de $\bar{x}_M$ selon l'itération M.Ensuite, nous avons utilisé la basée algorithme basée sur ADMM 7 de voir comment la robustesse du réseau réagit. Enfin, il prétend que quand la valeur intermédiaire de $\bar{x}$ est aussi proche que possible de la valeur réelle, l'indice de la robustesse du réseau peut être pris environ.

Dans le cas de la valeur de consensus inconnue $\bar{x}$, nous avons considéré le problème d'optimisation convexe qui recourt à une combinaison convexe de deux fonctions convexes. Le premier prend en compte le problème du consensus de moyenne, tandis que le second concerne l'estimation des valeurs propres du laplacien. Pour les grands graphes, une telle solution est évidemment insolubles. Conséquent, les Algorithmes 5, 6, 7, 8 sont efficaces et bien adaptées pour les réseaux de taille moyenn.

L'identification de la topologie du réseau se réfère à détecter et identifier les éléments du réseau intéressés et la relation entre les éléments du réseau destinataire et représente la construction de la topologie sous une forme appropriée. Conséquent, l'identification des réseaux de systèmes devient une tâche de plus en plus attrayante pour résoudre de nombreux problèmes dans différents domaines scientifiques et techniques.

En théorie, la matrice de consensus $\mathbf{W}$ admet la décomposition des valeurs propres $\mathbf{W} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, où $\mathbf{\Lambda} = \mathbf{I}_N - \epsilon\mathbf{\Delta}$, avec $\mathbf{\Delta}$ est matrice diagonale de $\lambda_i(\mathbf{L})$, représente la matrice diagonale des valeurs propres et $\mathbf{U}$ la matrice de vecteurs propres. Nous

pouvons trouver un moyen d'obtenir $\mathbf{U}$. Puisque la spectre de Laplace $sp(\mathbf{L})$ peut être réalisée en utilisant le méthodes proposées dans le Chapitre 4, nous pouvons reconstruire la topologie du réseau.

B.6.2 Travaux en cours et à venir

1. Conception des Protocoles de consensus de moyenne en temps fini:
   (a) La dépendance de la vitesse de convergence sur la séquence d'apprentissage reste à étudier. Ainsi, la conception de séquences d'apprentissage optimales peut être étudiée.
   (b) La recherche sur la robustesse du consensus de moyenne en temps fini.
2. L'évaluation de la robustesse du réseau:
   (a) Nouveaux méthodes qui peuvent être appliquées pour les graphes de grande échelle.
   (b) Pour la méthode indirecte, dans la pratique, il peut rencontrer le problème mal conditionné lors de l'établissement sous-matrices $\mathbf{Q}_i$, c le problème mal conditionné lors de l'établissement sous-matrices $\mathbf{Q}_i$ qui conduit à perdre le rang de ces sous-matrices. Conséquent, on peut étudier dans la précondition de ces matrices pour conserver la précision des méthodes proposées.
3. Reconstruire la topologie du réseau:
   (a) Concevoir une nouvelle méthode qui traite avec le spectre de Laplace $sp(\mathbf{L})$ dont les valeurs propres ne sont pas toutes distinctes.
   (b) D'évaluer la robustesse de ces méthodes lorsque l'échange de données sont sujettes à des communications imparfaites.

Bibliography

[1] H.B. Acharya and M.G. Gouda. On the hardness of topology inference. In *proc. of 12th International Conference on Distributed Computing and Networking (ICDCN 2011)*, pages 251–262, Bangalore, India, January 2011.

[2] A. Aderhold, D. Husmeier, J. J. Lennon, M. Colin Beale, and V. Anne Smith. Hierarchical Bayesian models in ecology: Reconstructing species interaction networks from non-homogeneous species abundance data. *Ecological Informatics*, 11(0):55 – 64, 2012.

[3] R. Aragues, G. Shi, D. V. Dimarogonas, C. Sagues, and K. H. Johansson. Distributed algebraic connectivity estimation for adaptive event-triggered consensus. In *Proc. of American Control Conference (ACC)*, pages 32–37, June 2012.

[4] J.S. Baras and P. Hovareshti. Efficient and robust communication topologies for distributed decision making in networked systems. In *Proc. of the 48th IEEE Conference on Decision and Control (CDC), held jointly with the 28th Chinese Control Conference (CCC)*, pages 3751–3756, December 2009.

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

[6] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proc. of the 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05*, pages 2996–3000, December 2005.

[7] V.D. Blondel, J.M. Hendrickx, and J. N. Tsitsiklis. Continuous-time average-preserving opinion dynamics with opinion-dependent communications. *SIAM Journal on Control and Optimization,*, 48:5214–5240, 2010.

[8] D. Boley. Local linear convergence of ADMM on quadratic or linear programs. *SIAM Journal on Optimization*, 23(4):2183–2207, 2013.

[9] S. Bolognani, S. Del Favero, L. Schenato, and D. Varagnolo. Distributed sensor calibration and least-squares parameter identification in WSNs using consensus al-

gorithms. In *Proc. of 46th annual Allerton Conference*, pages 1191–1198, Allerton House, UIUC, Illinois, USA, 2008.

[10] V. Borkar and P.P. Varaiya. Asymptotic agreement in distributed estimation. *IEEE Transactions on Automatic Control*, 27(3):650–655, Jun 1982.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[12] A. E Brouwer, A. M Cohen, and A Neumaier. *Distance-Regular Graphs*. Springer, 1989.

[13] J. W. Chinneck. *Practical optimization: A gentle introduction*. Systems and Computer Engineering. Carleton University. Ottawa, Canada, 2012.

[14] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization*. John Wiley & Sons, third edition, 2012.

[15] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, Aug 2006.

[16] A. H. Dekker and B. D. Colbert. Network robustness and graph topology. In *Proceedings of the 27th Australasian Conference on Computer Science (ACSC) - Volume 26*, 2004.

[17] D.V. Dimarogonas and K.J. Kyriakopoulos. Formation control and collision avoidance for multi-agent systems and a connection between formation infeasibility and flocking behavior. In *Proc. of the 44th IEEE Conference on Decision and Control and European Control Conference. (CDC-ECC)*, pages 84–89, Dec 2005.

[18] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.

[19] I. Shames E. Ghadimi, A. Teixeira and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers ADMM: quadratic problems. *IEEE Transactions on Automatic Control*, PP(99), 2014.

[20] W. Ellens and R. E. Kooij. Graph measures and network robustness. *e-version, arXiv*, 2013.

[21] W. Ellens, F.M. Spieksma, P. Van Mieghem, A. Jamakovic, and R.E. Kooij. Effective graph resistance. *Linear Algebra and its Applications*, 435(10):2491 – 2506, 2011. Special Issue in Honor of Dragos Cvetkovic.

[22] T. Erseghe. A distributed and scalable processing method based upon ADMM. *Signal Processing Letters, IEEE*, 19(9):563–566, September 2012.

[23] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista. Fast consensus by the alternating direction multipliers method. *IEEE Transactions on Signal Processing*, 59(11):5523–5537, 2011.

[24] A. Esna-Ashari, A.Y. Kibangou, and F. Garin. Distributed input and state estimation for linear discrete-time systems. In *Proc. 51st IEEE Conf. on Decision and Control (CDC)*, Maui, Hawaii, USA, December 2012.

[25] Ph. Ciblat F. Iutzeler, P. Bianchi and W. Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. Under review, 2014.

[26] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999.

[27] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu. Decentralized estimation of Laplacian eigenvalues in multi-agent systems. *Automatica*, 49(4):1031–1036, April 2013.

[28] M. Friedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[29] M.N. Vrahatis G.D. Magoulas and G.S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11:1769–1796, November 1999.

[30] L. Georgopoulos. *Definitive consensus for distributed data inference*. PhD thesis, EPFL, Lausanne, 2011.

[31] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. On the optimal step-size selection for the alternating direction method of multipliers. volume 3, pages 139–144, Santa Barbara, USA, 2012.

[32] C. Godsil and G. Royle. *Algebraic graph theory*. Springer, 2001.

[33] D. Hayden, Y. Yuan, and J. Goncalves. Network reconstruction from intrinsic noise: Minimum-phase systems. In *In proc. of American Control Conference (ACC)*, pages 4391–4396, June 2014.

[34] J.M. Hendrickx, R.M. Jungers, A. Olshevsky, and G. Vankeerberghen. Graph diameter, eigenvalues, and minimum-time consensus. *Automatica*, 50(2):635–640, 2014.

[35] J.M. Hendrickx, G. Shi, and K.H. Johansson. Finite-time consensus using stochastic matrices with positive diagonals. *IEEE Transactions on Automatic Control*, PP(99):1–1, 2014. doi:10.1109/TAC.2014.2352691.

[36] A. J. Hoffman and M. H. McAndrew. The polynomial of a directed graph. *Proc. American Mathematical Society*, 16:303–309, 1965.

[37] B. Li J. Ren, W.-Xu Wang and Y.-C. Lai. Noise bridges dynamical correlation and topology in coupled oscillator networks. *Physical Review Letters*, 104:058701, February 2010.

[38] A. Jadbabaie, Jie Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[39] A. Y. Kibangou. Step-size sequence design for finite-time average consensus in secure wireless sensor networks. *Systems and Control Letters*, 67:19–23, March 2014.

[40] A. Y. Kibangou and C. Commault. Decentralized Laplacian eigenvalues estimation and collaborative network topology identification. In *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys'12),Santa Barbara, USA*, pages 7–12, 2012.

[41] A.Y. Kibangou. Finite-time average consensus based protocol for distributed estimation over awgn channels. In *Proc. of the 50th IEEE Conference on Decision and Control (CDC)*, pages 261–265, Orlando, Fl, USA, December, 12-15 2011.

[42] A.Y. Kibangou. Graph Laplacian based matrix design for finite-time distributed average consensus. In *Proc. of the American Conference on Control(ACC)*, pages 261–265, Montreal, Canada, 2012.

[43] A.Y. Kibangou and C. Commault. Algebraic characterization of observability in distance-regular consensus networks. In *In 52nd IEEE Annual Conference on Decision and Control (CDC)*, pages 1313–1318, December 2013.

[44] A.Y. Kibangou and C. Commault. Observability in connected strongly regular graphs and distance regular graphs. *IEEE Transactions on Control of Network Systems*, 1(4):360–369, December 2014.

[45] A.Y. Kibangou and A.L.F. de Almeida. Distributed PARAFAC based DS-CDMA blind receiver for wireless sensor networks. In *Proc. of the IEEE Workshop SPAWC*, Marrakech, Morocco, June 20-23, 2010.

[46] C.K. Ko. *On matrix factorization and scheduling for finite-time average consensus.* PhD thesis, January, 2010.

[47] E. Kokiopoulou and P. Frossard. Polynomial filtering for fast convergence in distributed consensus. *IEEE Transactions on Signal Processing*, 57:342–354, January 2009.

[48] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[49] S. V. Macua, P. Belanovic, and S. Zazo. Consensus-based distributed principal component analysis in wireless sensor networks. In *Proc. of the IEEE Workshop SPAWC*, Marrakech, Morocco, June 20-23 2010.

[50] O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization (1994). *Optimzation Methods and Software*, 4:103–116, 1994.

[51] R. Merris. Laplacian matrices of a graph: a survey. *Linear Algebra and its Applications*, 197:143–176, 1994.

[52] M. Mesbahi and M. Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[53] E. Montijano, J.I. Montijano, and C. Sagues. Fast distributed consensus with Chebyshev polynomials. In *American Control Conference (ACC), 2011*, pages 5450–5455, June 2011.

[54] B.J. Moore and Carlos Canudas-de Wit. Formation control via distributed optimization of alignment error. In *Proc. of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference (CDC/CCC)*, pages 3075–3080, December 2009.

[55] F. Morbidi and A. Y. Kibangou. A distributed solution to the network reconstruction problem. *Systems & Control Letters*, 70:85 – 91, 2014.

[56] M. Nabi-Abdolyousefi and M. Mesbahi. Network identification via node knockout. *IEEE Transactions on Automatic Control*, 57(12):3214–3219, December 2012.

[57] A. Nedic, A. Ozdaglar, and P.A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[58] N. O'Clery, Y. Yuan, G.B. Stan, and M. Barahona. Observability and coarse graining of consensus dynamics through the external equitable partition. *Physical Review E*, 88:042805, October 2013.

[59] R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control,*, 51(3):401–420, March 2006.

[60] R. Olfati-Saber. Swarms on sphere: A programmable swarm with synchronous behaviors like oscillator networks. In *The 45th IEEE Conference on Decision and Control (CDC)*, pages 5060–5066, Dec 2006.

[61] R. Olfati-Saber. Distributed Kalman filtering for sensor networks. In *Proc. of the 46th IEEE Conf. on Decision and Control*, pages 5492–5498, New Orleans, LA, USA, December 12–14 2007.

[62] R. Olfati-Saber and R. M. Murray. Distributed cooperative control of multiple vehicle formations using structural potential functions. In *IFAC World Congress*, pages 346–352, 2002.

[63] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[64] R. Olfati-Saber and J.S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Proc. of the 44th IEEE Conference on Decision and Control and European Control Conference. (CDC-ECC)*, pages 6698–6703, December 2005.

[65] G. Parlangeli and G. Notarstefano. On the reachability and observability of path and cycle graphs. *IEEE Transactions on Automatic Control*, 57(3):743–748, March 2012.

[66] A. U. Raghunathan and S. D. Cairano. Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control. In *the 21st International Symposium on Mathematical Theory of Networks and Systems*, pages 807–811, Groningen, The Netherlands, 7-11 July 2014.

[67] T. Sahai, A. Speranzon, and A. Banaszuk. Hearing the cluster of a graph: A distributed algorithm. *Automatica*, 48(1):15–24, January 2012.

[68] S. Shahrampour and V. M. Preciado. Topology identification of directed dynamical networks via power spectral analysis. *IEEE Transactions on Automatic Control,*, 60(99):1–1, 2015.

[69] H. Shao and G. Zheng. Convergence analysis of a back-propagation algorithm with adaptive momentum. *Neurocomputing*, 74(5):749–752, 2011.

[70] S. Sundaram and C.N Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *Proc. of American Control Conference (ACC)*, pages 711–716, New York, USA, July 2007.

[71] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson. Optimal scaling of the ADMM algorithm for distributed quadratic programming. In *Proc. of the 52nd IEEE Conference on Decision and Control (CDC)*, pages 6868–6873, Florence, Italy, December 10-13, 2013.

[72] J. Wu, M. Barahona, Y. Tan, and H. Deng. Spectral measure of structural robustness in complex networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(6):1244–1252, November 2011.

[73] W. Wu. Convergence of gradient method with momentum for back-propagation neural networks. *Computational Mathematics*, 26(4):613–623, 2008.

[74] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems Control Letters*, 53:65–78, 2004.

[75] L. Xiao, S. Boyd, and S. Kim. Distributed average consensus with least-mean-square deviation. *Journal on Parallel Distributed Computation*, 67(1):33–46, January 2007.

[76] P. Yang, R. A. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity in mobile sensor networks. In *Proc. of American Control Conference,(ACC)*, pages 2678 – 2683, June 2008.

[77] P. Yang, R. A. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica 49*, pages 390–396, 2010.

[78] Y. Yuan, G. Stan, L. Shi, M. Barahona, and J. Goncalves. Decentralized minimim-time consensus. *Automatica 49*, pages 1227–1235, 2013.

[79] Y. Yuan, G. B. Stan, L. Shi, and J. Goncalves. Decentralised final value theorem for discrete-time LTI systems with application to minimal-time distributed consensus. In *Proc. of the 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. (CDC/CCC)*, pages 2664–2669, December 2009.