IMT School for Advanced Studies, Lucca

Lucca, Italy

Graph–based techniques and spectral graph theory in control and machine learning

PhD Program in Institutes, Markets and Technologies
Track in Computer, Decision and Systems Science
Curriculum in Control Systems

XXVIII Cycle

By

Rita Morisi

2016

The dissertation of Rita Morisi is approved.

Supervisor: Prof. Giorgio Gnecco, IMT School for Advanced Studies, Lucca

The dissertation of Rita Morisi has been reviewed by:

Prof. Franco Scarselli, Department of Information Engineering, University of Siena

Prof. Věra Kůrková, Department of Theoretical Computer Science, Academy of Sciences of the Czech Republic

IMT School for Advanced Studies, Lucca

2016

Contents

List of Figures

List of Tables

Vita

February 5, 1987	Born, San Giovanni in Persiceto (BO), Italy
2006–2009	B.SC in Matematics Mark: 110/110 cum laude University of Bologna Bologna, Italy
2009–2011	M.SC in Matematics Mark: 110/110 cum laude University of Bologna Bologna, Italy
2013–present	PhD candidate IMT School for Advanced Studies Lucca, Italy

Publications

Papers in international journals

- G. Gnecco, R. Morisi, G. Roth, M. Sanguineti, A. C. Taramasso, "Supervised and semi–supervised classifiers for the detection of flood–prone areas", Soft Computing, pp. 1–13, 2015

- G. Gnecco, R. Morisi, A. Bemporad, "Sparse solutions to the average consensus problem via various regularization of the fastest mixing Markov–chain problem", IEEE Transactions on Network Science and Engineering, vol. 2, n. 3, pp. 97–111, 2015

- R. Morisi, B. Donini, N. Lanconelli, J. Rosengarden, J. Morgan, S. Harden, N. Curzen, "Semi–automated Scar Detection in Delayed Enhanced Cardiac Magnetic Resonance Images", International Journal of Modern Physics C (IJMPC), vol. 26, n.1, 2015

- C. Rusu, R. Morisi, D. Boschetto, R. Dharmakumar, S. A. Tsaftaris, "Synthetic Generation of Myocardial Blood–Oxygen–Level–dependent MRI Time Series Via Structural Sparse Decomposition Modeling", IEEE Transactions on Medical Imaging, vol. 33, n. 7, pp. 1422–1433, 2014

- R. Morisi, G. Gnecco, N. Lanconelli, S. Zanigni, D. Manners, C. Testa, S. Evangelisti, L. L. Gramegna, C. Bianchini, P. Cortelli, C. Tonon, R. Lodi, "Binary and multi–class classification of parkinsonian disorders with support vector machines based on quantitative brain MR and graph–based features", Movement Disorders (submitted)

- Rita Morisi, Giorgio Gnecco, Alberto Bemporad "A hierarchical consensus method for the approximation of the consensus state, based on clustering and spectral graph theory", Engineering Applications of Artificial Intelligence (submitted)

Conference papers

- L. L. Gramegna, C. Testa, R. Morisi, S. Zanigni, G. Gnecco, N. Lanconelli, D. N. Manners, S. Evangelisti, P. Cortelli, C. Tonon, R. Lodi, "Binary and multi–class classification of parkinsonian disorders with support vector machines based on quantitative brain MR and graph–based features", ISMRM Italian Chapter, 2016

- G. Gnecco, A. Bemporad, R. Morisi, M. Gori, M. Sanguineti, "Online learning as an LQG optimal control problem with random matrices", Proceedings of IEEE ECC 2015

- R. Morisi, G. Gnecco, N. Lanconelli, et al., "Binary and multi–class Parkinsonian disorders classification using Support Vector Machines", In: Lecture Notes in Computer Science, Springer, 2015

- G. Gnecco, R. Morisi, and A. Bemporad, "Sparse Solutions to the Average Consensus Problem via l_1–norm Regularization of the Fastest Mixing Markov–Chain Problem", In: Proceedings of the 53rd Annual Conference on Decision and Control (CDC), IEEE, pp. 2228-2233., ISBN 978-1-4799-7746-8, 2014

- L. Lara-Rodriguez, S. Vera, F. Perez, N. Lanconelli, R. Morisi, B.Donini, et al., "Supervised Learning Modelization and Segmentation of Cardiac Scar in Delayed Enhanced MRI". Lecture Notes in Computer Science, vol. 7746, 2013, Statistical Atlases and Computational Models of the Heart, Imaging and Modelling Challenges

- L. Lara-Rodriguez, S. Vera, F. Perez, N. Lanconelli, R. Morisi, et al., "Cardiac scar detection, segmentation and quantification in MRI images for ICD treatment planning", International Journal of Computer Assisted Radiology and Surgery, vol. 7, s. 1 (Proceedings of CARS 2012, Pisa, Italy), 2012.

- R. Morisi, R. Dharmakumar, and S. A. Tsaftaris, "Unsupervised Ischemia Detection at Rest with CP–BOLD Cardiac MRI: A Simulation Study Employing Independent Component Analysis", Presented for the International Society of Magnetic Resonance in Medicine (ISMRM) Meeting, Milan, May 2014, Magna Cum Laude Award received

Abstract

Graphs are powerful data structure for representing objects and their relationships. They are extremely useful in the study of dynamical systems, evaluating how different agents interact among each other and behave. An example is represented by the consensus problem where a graph models a set of agents that locally interact and exchange their opinions with the aim of reaching a common opinion (consensus state). At the same time, many learning techniques rely on graphs exploiting their potentialities in modeling the relationships between data and determining additional features related to the data similarities. To study both the consensus problem and specific machine learning applications based on graphs, the study of the spectral properties of graphs reveals fundamental. In the consensus problem, the convergence rate to the consensus state strictly depends on the spectral properties of the transition probability matrix associated to the agents network. Whereas graphs and their spectral properties are fundamental in determining learning algorithms able to capture the structure of a dataset. We propose a theoretical and numerical study of the spectral properties of a network of agents that interact with the aim of increasing the rate of convergence to the consensus state keeping as sparse as possible the graph involved. Experimental results demonstrate the capability of the proposed approach in reaching the consensus state faster than a classical approach. We then investigate the potentialities of graphs when applied in classification problems. The results achieved highlight the importance of graphs and their spectral properties handling with both semi–supervised and supervised learning problems.

Part I

Introduction and background

1 Introduction

1.1 Graphs versatility and applications

Graphs are a general and powerful data structure for objects representation; they are used to model entities, their attributes and their relationships (1). Generally speaking, a graph is characterized by a set of nodes, which represent the different entities, and a set of edges that link pairs of nodes and represent the interconnections between the nodes. In particular, a graph is directed if the edges have a direction associated with them, undirected otherwise. Nowadays, graphs are becoming increasingly important in modeling structures and their interactions, with broad applications including computer vision, bioinformatics, text retrieval, and Web analysis, where they represent a useful tool for searching and for community discovery (2). In modeling physical and biological processes (3; 4; 5) graphs, for instance, are used to represent connections between interacting parts of a system, to model the dynamics of a physical process, disease propagation, as well as the evolution of populations in an ecosystem (6). Several biological systems, in fact, can be usefully represented as networks. Examples are the genetic regulatory network and the food web network, which can be modeled by a graph with nodes representing species in an ecosystem and directed edges in-

dicating which species prey on the other (3). In addition, they are widely used in the study of social systems (7), where a social network is a set of people or groups of people with some patterns of contacts or interactions between them (8). The patterns of friendships between individuals, business relationships between companies are examples of social networks (3). At the same time, graphs represent a useful tool to study the flow of information between different systems (9), how news and information spread between single individuals and groups of agents (10; 11), and to analyze the flow of traffic on roads (10). Finally, another application of graphs is represented by the network of citations between academic papers. In this case, the nodes of the network are the articles, while the directed edges indicate which articles cite the others.

At the same time, in the machine learning field different methods rely on graphs. There are both unsupervised and supervised learning techniques that make use of graphs. Regarding unsupervised and semi–supervised methods, graphs can provide a model of the manifold where the data lie on and they are also used to determine clusters of similar data (12). In a supervised context, graphs can be exploited to infer labels or numerical values attached to nodes, to extract novel and useful knowledge from data, and additional features (13). In addition, they can be also combined with other learning techniques, such as the so–called kernel methods (14), providing a useful tool to deal with several learning problems.

Moreover, in (15) the idea of constructing kernels on the nodes of graphs was first proposed, with the aim of capturing the relationships between data points induced by the local structure of the graph; the work (16), instead, focuses on kernels capturing the similarity of whole graphs (that idea was first proposed by (17)). For both these types of graph kernels, the challenge is to define a kernel that captures the semantics inherent in the graph structure (16). Examples of application of graph kernels can be found, for instance, in (18), where they are used to compare biological networks, or in (19), for the prediction of protein function.

Finally, another application of graphs in the field of patter recognition and computer vision is represented by the graph matching problem

(20; 21). This problem consists in searching for an edge–preserving mapping between the nodes of two graphs (22). Specifically, in the field of computer vision, many problems are formulated as an attributed graph matching problem; the nodes of the graphs correspond to local features of the image and edges correspond to relational aspects between features. The final goal is to find a correspondence between nodes of the two graphs such that they "look similar" (23). In a pattern recognition context, instead, the graph matching problem can be found, for instance, in the human faces recognition problem (24).
Beside the problems already introduced, additional learning problems that rely on graphs concern, for instance, the problems of character recognition, shape analysis (25), Web document analysis (26; 27; 28) and data mining (29).

1.2 Problem statement

One of the problems investigated in the theory of dynamical systems is the consensus problem. This kind of situation is characterized by a group of agents that locally interact and exchange opinions. They are modeled by a graph with nodes corresponding to the agents and edges representing their interconnections. For instance, if the graph of interconnections is undirected, an edge between two different agents (i.e., nodes) represents the possibility for one of the two to receive information from the other and vice–versa (30; 31). Agents influence each other depending on the strength of the interconnections between them (32). In particular, a "consensus algorithm" is an interaction rule that specifies the information exchange between an agent and all of its neighbors on the network (33). The final goal of this problem is to have all the agents agree upon a common opinion, a "consensus", that means to reach an agreement regarding a certain quantity of interest that depends on the state of all agents. Several applications of this kind of problem exist. For instance, the consensus problem arises when dealing with the collective behavior of networked agents (34), in the context of multiagent coordination (35; 36), sensor networks (37) and many other applications (38).

At the same time different consensus algorithms and studies have been developed (39; 40; 41).

In this dissertation we focus on the average consensus problem, where the graph of interconnections is undirected (i.e., the edges are not orientated, thus an agent connected to another can communicate with him and vice–versa), and the final consensus state is given by the average of their initial opinions (42). In particular we are interested in studying the convergence rate to the consensus state. In this context, the spectral properties of the network which models the group of agents play a crucial role. In particular, the Laplacian matrix and its spectral properties (43; 44) are fundamental to study how the network of agents evolves. Studies related to the convergence rate to the consensus state have been developed, for instance, in (45; 46; 47), while in (48) it is studied the related problem of the mixing time.

We intend to focus from both a theoretical and practical point of view on the spectral properties of the network involved in order to increase the convergence rate to the consensus state keeping, at the same time, as sparse as possible the agents network. We require, in fact, to obtain solutions able to keep the cost of communications between the different agents small.

On the other hand, in the thesis the spectral properties of graphs are also exploited in a machine learning context. Beside the application of graphs to the study of dynamical systems and of interconnections of individuals, we intend to study the potentiality of graphs in their application to classification problems. In particular, we aim at analyzing the spectral properties of graphs, previously evaluated in the consensus problem, in a semi–supervised learning context. We focus on the application of a semi–supervised learning technique that relies on a graph built on the dataset for the classification of flood–prone areas. Graphs, in situations where both labeled and unlabeled data are present, represent a fundamental tool to handle with both such types of data, making possible to extract additional information from the unlabeled data. The spectral properties of graphs, in this context, reveal to be fundamental to estimate the data probability distribution. This knowledge is fundamental to compute a

classifier able to usually achieve better performance than a classical one that does not rely on the spectral properties of a graph adequately computed on the dataset.

Finally, the potentialities of graphs in a learning context are shown in another classification problem. We study and apply graphs in a classification problem of parkinsonian disorders in order to model the dataset, extracting additional graph–based features from the data. We aim at computing features that capture the similarities between data belonging to the same class and the dissimilarities between data coming from different classes. The comparison between this approach and a classical one that does not rely on graphs highlights again the capabilities of graphs in modeling data and extracting additional and useful information.

1.3 Challenges and scientific contributions

The main contribution of this dissertation relies on the theoretical and practical studies of graphs and their spectral properties in both the control field and in machine learning problems. In Part II we present original theoretical studies related to the solution of the consensus problem and we provide theoretical and practical solutions able to sparsify the network of the agents, keeping at the same time the rate of convergence to the consensus state sufficiently high. We then focus on a novel approach able to increase the convergence rate to the consensus state dividing the original network in several subgraphs. Again, we ground our study on graph spectral theory and in particular on the application of the Cheeger's inequality (49).

Beside our investigation of the consensus problem, in Part III, we also provide novel contributions in specific classification problems. We highlight the potentiality of a semi–supervised learning technique based on graphs in improving the classification performance of a classical supervised classifier that does not exploit spectral properties of graphs. Finally, we conclude our study on graphs evaluating their capabilities in modeling a dataset and extracting additional information from it.

After an overview of graphs and their basic notations, in **Chapter**

2 we report and describe fundamental graph features that are considered in the following chapters. In particular we briefly introduce spectral graph theory and its notions. The study of the eigenvalues and eigenvectors of the matrices related to graphs represents the common denominator between the two different topics treated in this dissertation.

In **Chapter 3** the consensus problem is presented. We introduce the problem of dealing with a large number of agents that locally interact and have to reach a common opinion as fast as possible. We focus on the study of the graph topology determining the structure of the network that leads to the fastest convergence rate to the consensus state. In particular, we are interested in determining a sparse variation of the problem, with the aim of determining a sparse graph with a fast convergence rate to the consensus state. Dealing with large networks is, from a computational point of view, expensive and time consuming especially when huge networks are considered. Hence, determining a method able to provide satisfactory results in terms of convergence rate to the consensus state maintaining, at the same time, the graph as sparse as possible is useful and advantageous in many applications where large networks are considered. In **Chapter 4** we continue the study on the consensus problem and the rate of convergence to the consensus state. Differently from the previous chapter, where a convex optimization problem is considered, now the idea is to opportunely divide the original network in different subnetworks (i.e., subsets of agents), having fast convergence rate to their local consensus state. We thus decompose the consensus problem in many consensus subproblems and then we investigate the final consensus problem "merging" the local consensus states previously computed. A suitable technique able to extract from the original graph different subgraphs where the convergence rate to their own consensus state is fast, reveals useful to increase the convergence rate to the consensus state computed on the original network, without using such technique. In **Chapter 5** the spectral properties of graphs are again studied but they are applied in a different context. A machine learning problem is considered, dealing with a semi–supervised classification problem, where both labeled and unlabeled data are provided to a classifier. In particu-

lar, we apply a semi–supervised learning technique for the classification of flood–prone areas. The main purpose of the specific study is the evaluation of the capabilities of a semi–supervised learning technique that exploits the spectral properties of a graph used to model the manifold where the data lie on. We compare the results achieved by the semi–supervised classifier with the ones achieved by a supervised learning technique trained on the labeled data only. A graph–based learning technique reveals suitable for this particular classification context. Finally, in **Chapter 6** we show an additional application of graphs in the machine learning field. Again a classification problem is presented, dealing with different parkinsonian disorders. Contrary to the previous problem, in this case graphs are not used in the learning algorithm, but they provide a model of the dataset highlighting the similarities and dissimilarities between the data. A graph is built on the dataset with the aim of extracting additional features from the data that have to be classified. We observe that graphs are a powerful structure able to provide useful information to improve the classification results achieved when graph–based features are not added to the dataset. Finally, **Chapter 7** offers a final discussion and the concluding remarks highlighting the important developments achieved in this work and future improvements to carry out.

2 Introduction to graphs and spectral properties

2.1 Introduction

In this preliminary chapter we provide a brief introduction to graphs. We introduce the basic notations and information necessary to deal with the arguments described later. We first recall the generic notations used to introduce and describe graphs, then a selection of features related to both single nodes and subsets of nodes in a graph will be provided. We introduce only the notations that will be recalled and used in the following chapters. Subsequently, we will focus on spectral graph theory whose properties will be discussed in Chapters 3 and 4, where the consensus problem is studied, and Chapter 5 where semi–supervised learning techniques are studied and applied. In particular, we will focus on the study of the eigenvalues of the Laplacian matrix of a graph introducing important results that will be used and considered in the study of the consensus problem.

2.2 Graph representation

A graph G consists of a finite set of nodes (or equivalently vertexes) V and a set of edges E; its representation is given by a pair $G = (V, E)$. From now on, we will indicate with v_i a generic node in the vertex set V, while with $e_{ij} \in E$ we indicate the edge between a couple of nodes v_i and v_j. $|V|$ indicates the cardinality of V; thus, if $V = \{v_1, v_2, \ldots, v_N\}$, it follows that $|V| = N$. We are interested in undirected graphs where no orientation in the edges is present, while weighted and unweighted graphs will be both taken in consideration. The difference between the two is related to the presence or not of a weight associated to the edges between the different pairs of nodes. We will use the same notation to indicate both the unweighted and weighted graphs. For the latter we will sometimes indicate with W the weighted adjacency matrix (used extensively in the following chapters) whose entries w_{ij} specify the weight associated to each edge e_{ij}. The weight w_{ij} is a real non negative number that represents the strength of the connection between the pair of nodes v_i and v_j, $\forall\, i, j = 1, \ldots, N$. Note that, for undirected graphs, $w_{ij} = w_{ji}$.

As just introduced for weighted graphs, we recall that both a weighted and an unweighted graph can be entirely represented by a matrix A which is called the adjacency matrix. If $|V| = N$, the adjacency matrix A is an $N \times N$ matrix whose entries are defined as

$$a_{ij} = \begin{cases} w_{ij} & \text{if } (v_i, v_j) \text{ are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

in the case of weighted graphs. Sometimes we will use either A or W to indicate the adjacency matrix in the weighted case. If unweighted graphs are considered, the adjacency matrix is a binary matrix A with entries:

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

Again, if the graph is undirected, the adjacency matrix A is symmetric.

Note that, in this introductory chapter, when defining graph features and important quantities related to graphs, we do not consider nodes

with self–loops. Anyway, in Chapters 3 and 4 where the consensus problem is studied, we will introduce graphs presenting self–loops.

In the following section we first provide a description of a series of graph features (50; 51; 52; 53) that will be used in the following chapters.

2.2.1 Graphs features and fundamental notions

Nodes definitions and features

Let G be a generic graph, with N nodes, i.e., $|V| = N$; G could be either a weighted graph or an unweighted one. We now report some measures of the importance of a generic node $v_i \in G$; indices that define the importance of a node are the so–called centrality measures (6; 54). First, we mention the degree of a node v_i, with $i = 1, \ldots, N$, which is defined as $d_i = \sum_{j=i}^{N} a_{ij}$. It is the sum of the edge values incident to the vertex, where each of these edges has value 1 in the unweighted case, w_{ij} otherwise. The sum of the degrees of all the vertices is the total degree of a graph. The degree matrix D of a graph, instead, is defined as the diagonal matrix with the degrees $d_1, \ldots, d_N$ on the main diagonal. In general an important node is involved in a large number of interactions.

Beside the degree of a node, before introducing other centrality measures, we need to introduce the natural distance metric between pairs of nodes; this measure is the length of the shortest path between two nodes. More precisely, given two nodes v_i and v_j, a shortest path between them is defined as any of the paths (i.e., a sequence of edges which connect the two nodes) with the minimum number of edges (in the unweighted case) or with the minimum sum of the weights of its constituent edges, for weighted graphs. Thus, as previously mentioned, the distance between pairs of nodes is measured as the length of any of the shortest paths between them, i.e., the sum of the weights of the edges in such shortest path. We define the length of the shortest path between nodes v_i and v_j as $\delta(v_i, v_j)$.

Important centrality measures are:

- closeness centrality: given a node v_i, it is defined as

$$\mathrm{C_c}(v_i) := \frac{1}{\sum_{v_j} \delta(v_i, v_j)}.$$

  This feature defines the importance of a node measuring if this node is "close" to and can communicate quickly with the other nodes in the graph;

- betweennes centrality: given a node v_i in G, this centrality measure is defined in the following way:

$$\mathrm{B_c}(v_i) := \sum_{v_j \neq v_i \neq v_k \in V} \frac{\sigma_{v_j, v_k}(v_i)}{\sigma_{v_j, v_k}},$$

  where σ_{v_j, v_k} is the total number of shortest paths from node v_j and node v_k, while $\sigma_{v_j, v_k}(v_i)$ is the number of those paths that pass through v_i. This measure tells us that an important node lies on a high proportion of shortest paths between other nodes.

We then mention the vertex eccentricity $e(v_i)$ of a generic node v_i defined as $e(v_i) = \max\{\delta(v_i, v_j) : v_j \in V\}$; the radius $r(G)$ of a graph that is $r(G) = \min\{e(v_i) : v_i \in V\}$. Finally, we introduce the clustering coefficient of a node v_i that is a measure of the degree to which nodes in a graph tend to cluster together. More precisely, defining the neighborhood N_i of a vertex v_i as the set made of its immediately connected neighbors, i.e., $N_i = \{v_j : e_{ij} \in E\}$, the clustering coefficient $C(v_i)$ of node v_i is defined as:

$$C(v_i) = \frac{2|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)},$$

where k_i is the number of neighbors of node v_i.

Subgraphs notions

From the vertex set V of a graph, it is possible to consider a generic subset of vertices $S \subseteq V$ that define a subgraph of the original graph G. In

particular, given the subset $S \subseteq V$, we denote its complement as $V \setminus S$ or, equivalently as $\bar{S}$. The "size" of subset S is defined either as

$$|S| := \text{the number of vertices in S},$$

or as

$$\text{vol}(S) := \sum_{i \in S} d_i. \tag{2.1}$$

In particular, we say that a subset S is connected if any two vertices in S can be joined by a path such that all intermediate nodes also lie in S. Moreover, a subset S is called a connected component if it is connected and if there are no connections between nodes in S and $\bar{S}$. Finally, given two disjoint subsets $S, T \subset V$, we define the *cut* between them as:

$$\text{cut}(S, T) := \sum_{i \in S, j \in T} w_{ij} \tag{2.2}$$

Thus, the *cut* between the sets of nodes S and T is the sum of the edge values with an extremity in S and another in T. Finally, we introduce the definition of connected graph that will be extensively used in Chapters 3 and 4. An undirected graph G is connected if each pair of vertices forms the endpoints of a path (i.e., for each v_i and v_j nodes in G there is a path which connects them). In addition, when dealing with directed graphs, a directed graph G is said to be strongly connected if for each two vertices there are paths from one to the other in both directions. The notations just introduced are necessary to deal with the consensus problem and the particular development described in Chapter 4.

Before diving into spectral graph theory and its properties, we describe some methods used to build a graph that will be considered in Chapters 5 and 6.

2.2.2 Similarity graphs

Given a dataset $X = \{x_1, \ldots, x_N\}$, where $x_i \in \mathbb{R}^n$, with $i = 1, \ldots, N$, are n–dimensional vectors, if one is interested in building a graph $G = (V, E)$ on these data points, either a weighted one or an unweighted one,

it is necessary to define a notion of similarity $s_{ij} \geq 0$ between all pairs of data x_i and x_j. More precisely, each data point x_i corresponds to a node v_i, while the similarity measure s_{ij} encodes the strength of the connection between points x_i and x_j. In general, these two points are connected by an edge e_{ij} if the similarity measures s_{ij} is positive or larger than a certain threshold. We use the term similarity measure, or equivalently, distance measure to describe three techniques commonly used to define a graph starting from the points; they are:

- *ϵ–neighborhood graph*: the points connected are those with distances s_{ij}, with $i = 1, \ldots, N$, smaller than a given $\epsilon > 0$;

- *k-nearest neighbor graph*: for a given k, a vertex v_j, corresponding to an element x_j, is connected to another vertex v_i if v_j is among the k nearest neighbors of v_i. Since we will deal with only undirected graphs, and the one built following this rule is a directed one, one of the following ways to make the graph undirected can be applied. Either v_i is connected to v_j if v_i is among the k–nearest neighbors of v_j *or* vice–versa, determining a *k–nearest neighbor graph*; or v_i and v_j are connected by an edge if v_i is among the k–nearest neighbors of v_j *and* v_j is among the k–nearest neighbors of v_i. The resulting graph is called *mutual k–nearest neighbor graph*: this construction leads to a sparser graph than the previous one. Both unweighted and weighted *k-nearest neighbor graphs/mutual k–nearest neighbor graphs* can be built depending on the choice of assigning the value 1 to the edges connecting the nodes or a weight related to the distance between the pairs of nodes;

- *fully connected graph*: between all the possible pairs of points the Gaussian similarity function $s_{ij} = \exp(\frac{\|x_i - x_j\|^2}{2\sigma^2})$ is computed determining the weighted matrix W with entries $w_{ij} = s_{ij}$.

From the similarity measures, depending on the particular application and purpose one is dealing with, either a weighted graph can be obtained by using as edge weights the similarity measures themselves or an unweighted graph can be built. It can be obtained by threshold-

ing the values of the weights and assigning an edge with value 1 only between pairs of points x_i and x_j with similarity s_{ij} that satisfies certain requirements. Otherwise, if a *k–nearest neighbor graph* is considered, it can be transformed into an unweighted one by assigning value 1 to the edges instead of their original weight (55).

We have introduced a brief overview on graph notations useful for the studies presented in the next chapters. We now continue the introduction to graphs focusing on spectral graph theory and its properties. Our main goal is to introduce definitions and techniques able to determine clusters inside a graph. In Chapter 4, in fact, we will need to determine clusters inside a single graph in order to increase the rate of convergence to the consensus state.

2.3 Spectral graph theory

This branch of graph theory is concerned with the analysis of graphs by using algebraic properties of associated matrices. In particular, it studies the relation between graph properties and the spectrum of the adjacency matrix or the Laplacian matrix (56). The latter matrix is defined as $L = D - A$, where D is the degree matrix of a graph, while A is either the binary adjacency matrix in the case of unweighted graphs or the weighted adjacency matrix if weighted graphs are considered. More specifically, L is the matrix with components equal to:

$$L_{ij} = \begin{cases} d_i & \text{if i = j} \\ -1 & \text{if } v_i \text{ and } v_j \text{ are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

For our purposes we denote with ξ_i, with $i = 0, \ldots, N-1$ the eigenvalues of the Laplacian matrix, where, in the following, the eigenvalues are considered in a non–decreasing order, i.e., $\xi_0 \leq \xi_1, \ldots, \leq \xi_{N-1}$. Note that L, defined as above, is also called unnormalized Laplacian matrix; in the following we list some properties of L that can be found, for instance in (55; 57; 58).

Remark 1. *The matrix L satisfies the following properties:*

- $\forall x \in \mathbb{R}^N$ *one has* $x^T L x = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij}(x_i - x_j)^2$ *;*
- L *is a symmetric positive-semi-definite matrix;*
- *the smallest eigenvalue of* L *is* $\xi_0 = 0$*. Moreover, the indicator vector* $\mathbf{1} = (1, \cdots, 1)^T$ *is an eigenvector associated with the eigenvalue* 0*;*
- *from the second property it follows that the eigenvalues of* L *are real and non–negative values*

$$0 = \xi_0 \leq \xi_1, \ldots, \leq \xi_{N-1}.$$

Another important property of L that is fundamental in spectral graph theory is the following:

Remark 2. *The multiplicity of the eigenvalue* 0 *of* L *is equal to the number of connected components of the graph, and a basis of the associated eigenspace is made of the indicator vectors of such connected components.*

We refer to (55) for the proof of the propositions stated above.

In many applications two possible variations can be used instead of the unnormalized Laplacian matrix L. They are:

- the normalized Laplacian matrix given by $L_N = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$,
- the random walk Laplacian matrix defined as $L_{rw} = I - D^{-1} A$.

Where, again, A can be either a binary or a weighted adjacency matrix. If graphs with isolated nodes are considered (i.e., a graph G with at least a node v_i with degree $d_i = 0$), we introduce the following notation reported in (59):

$$d_{ii}^{-1} = d_{ii}^{-\frac{1}{2}} = 0 \; \forall \, v_i \text{ such that } d_i = 0$$

In particular, the structure of matrix L_N is the following:

$$L_{N\,ij} = \begin{cases} 1 & \text{if } d_i \neq 0 \text{ and } i = j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{if } (v_i, v_j) \text{ are connected by an edge and } d_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

while

$$L_{rwij} = \begin{cases} 1 & \text{if } d_i \neq 0 \text{ and } i = j \\ -\frac{1}{d_i} & \text{if } (v_i, v_j) \text{ are connected by an edge and } d_i \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Similar properties like the ones listed in Remarks 1 and 2 hold for the two versions of the normalized Laplacian matrices, that can be found, for instance, in (49; 60; 61; 62). In the following we will indicate with ξ_i the eigenvalues of both the unnormalized and the normalized Laplacian matrix, depending on the application we will consider.

Now, our purpose is twofold. We first intend to introduce some fundamental notions of spectral clustering, because, as we will describe in Chapter 4, we need a method able to determine clusters inside a graph in order to increase the convergence rate to the consensus state. Then, we aim at introducing some important quantities related to the set of eigenvalues of L in order to better understand the topology of a given graph (49). Of course, these two tasks are strongly related.

2.3.1 Spectral clustering technique

The goal of spectral clustering is to cluster data making use of the eigenvalues and eigenvectors of the Laplacian matrix. In particular, for spectral clustering algorithms, both the unnormalized version and the normalized version of the Laplacian matrix can be used. All the spectral clustering algorithms are able to cluster a set of data points $X = \{x_1, \ldots, x_N\}$ by means of the eigenvectors of the Laplacian. More precisely, as stated in Remark 2, by considering the multiplicity of the eigenvalue 0 of the Laplacian matrix, it is possible to estimate the number of clusters inside a graph. In fact, as shown in (63), if a connected component has a structure with k "apparent" clusters, the first k eigenvalues will be close to 0, while starting from the $k+1$-th eigenvalue, their value will be significantly larger than 0. In this case, the projection of the data points x_i on the subspace defined by the first k eigenvectors corresponding to the k smallest eigenvalues ξ_j, with $j = 0, \ldots, k-1$, makes the clustering process easier to be performed. In particular, without going

into details, a spectral clustering algorithm, starting from the similarity graph W and the number k of clusters that are supposed to be present in the original graph, computes the first k eigenvectors $u_0, \ldots, u_{k-1}$ of the unnormalized or normalized Laplacian matrix. It subsequently builds a matrix $U \in \mathbb{R}^{N \times k}$ containing the vectors $u_0, \ldots, u_{k-1}$ as columns. Finally, the algorithm clusters the new points corresponding to the rows of U (that are the original nodes projected in the new k–dimensional space spanned by the first k eigenvectors of the Laplacian matrix), with the k–means clustering algorithm (64).

The spectral clustering technique can be derived starting from different theories and points of view. In the following section we briefly list some of these theories and the connection with spectral clustering. For further information and for the proofs of the statements we will introduce, the reader is referred to (55; 65).

2.3.2 Spectral clustering points of view

Let $X = \{x_1, \ldots, x_N\}$ be a set of data points and let G the corresponding similarity graph G with adjacency matrix W built following one of the methods described in Section 2.2.2. Spectral clustering can be explained by the following theories.

Graph cut point of view

As previously mentioned, clustering tends to separate points in different groups according to their similarities; it means that the goal of this kind of method is to partition a graph such that the edges between different groups have a very low weight, while edges inside the same cluster have high weight. This problem can be translated in solving the mincut problem. More precisely, starting from the definition of *cut* between two different subsets (Formula (2.2)), given a generic number k of subsets, the mincut problem consists in determining the partition $S_1, S_2, \ldots, S_k$ that minimizes

$$\text{cut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \text{cut}(S_i, \overline{S_i}). \tag{2.3}$$

Thus, the minimization of (2.3) imposes the edges between different groups to have a very small weight. However, since the solution of problem (2.3) often leads to unsatisfactory results, in general, two different functions are preferred:

$$\text{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S_i})}{|S_i|}; \tag{2.4}$$

$$\text{Ncut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S_i})}{\text{vol}(S_i)}. \tag{2.5}$$

These two different functions are usually preferred since in many cases, the minimization of (2.3) determines a solution given by one individual vertex separated from the rest of the graph. Thus, in order to have reasonably large clusters, minimizing either equation (2.4) or (2.5) which are normalized by the size of the subset, leads to a solution with reasonably large subsets $S_1, S_2, \ldots, S_k$. In particular, the RatioCut equation measures the size of a subset S_i by its number of vertices $|S_i|$, while the Ncut equation divides the *cut* by the weight $\text{vol}(S_i)$ of the edges in subset S_i. Now, it is possible to prove that spectral clustering represents a way to solve a relaxed versions of the RatioCut problem (Formula (2.4)) and the Normalized cut problem (Formula (2.5)) (55; 60; 61; 66).

Random walks point of view

Random walks on the similarity graph can be used to derive another possible explanation of spectral clustering. In particular, spectral clustering can be associated to the problem of finding a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters.

Defining by p_{ij} the transition probability of jumping in one step from vertex v_i to vertex v_j, it is common to assume that p_{ij} is proportional

to the edge weight w_{ij}; in particular, a possible choice is $p_{ij} := \frac{w_{ij}}{d_i}$. It follows that the transition matrix $P = (p_{ij})_{i,j=1,\dots,N}$ of the random walks is defined by

$$P = D^{-1}W,$$

that is strictly related to the normalized Laplacian, since $L_{rw} = I - P$. In particular, this equivalence will be discussed in details in Chapter 3 and 4.

Now, as (67) shows there is a formal equivalence between the Ncut and the transition probabilities of random walks; thus, solving the Ncut minimization problem allows to determine a cut through the graph such that a random walk seldom passes through a cluster and another. This explains another spectral clustering point of view.

Matrix perturbation point of view

The last interpretation of spectral clustering derives from perturbation theory (68; 69). This studies how eigenvalues and eigenvectors of a matrix change if a small perturbation is added to the matrix itself. In particular, the formal basis for the perturbation approach to spectral clustering is the Davis–Kahan theorem (70).

Now, if we consider a disconnected graph with k connected components, we know that the first k eigenvectors of L or L_{rw} are the indicator vectors of the clusters. If we then slightly perturb (i.e., by adding a small number of edges between nodes in different clusters) the original graph, the corresponding perturbed (normalized or unnormalized) Laplacian is a small perturbation of the original Laplacian matrix. Now, from Davis–Kahan theorem it follows that the relationship between matrix perturbation theory and spectral clustering resides on the choice of an appropriate interval I that contains both the first k eigenvalues of the Laplacian in the ideal case (disconnected graph with k connected components) and the ones of the perturbed Laplacian matrix. In particular, the choice of the appropriate interval I is easier the smaller the perturbation of L and the larger the eigengap $|\xi_{k+1} - \xi_k|$. Finally, from Davis–Kahan theorem it follows that the larger the eigengap, the closer the eigenvectors of the

ideal case and the perturbed case, and hence the better spectral clustering works. For further details we refer to (55; 70).

In the next section a detailed study on the second smallest eigenvalue ξ_1 of the normalized Laplacian matrix L_N is carried out. This eigenvalue, in fact, is an indicator of the rate of convergence to the consensus state, argument that will be studied in the following two chapters.

2.3.3 Laplacian eigenvalues and their properties

In this section, we focus on the study of the Cheeger's inequality, (see, e.g., (71)) which provides an upper and lower bound for the second smallest eigenvalue ξ_1 of the normalized Laplacian matrix L_N.

The Cheeger's inequality, as described in (71) relates the eigenvalues of the normalized Laplacian matrix associated to a graph with the Cheeger's constant of the graph. In particular, if $G = (V, E)$ is a graph and $S \subset V$, denoting with $\partial(S)$ the boundary of S, i.e.,

$$\partial(S) := \{(u, v) \in E : u \in S, v \notin S\}$$

the Cheeger's constant is defined in the following way:

$$\Phi(G) := \min_{S \subset V} \frac{|\partial(S)|}{\min\{\mathrm{vol}(S), \mathrm{vol}(V - S)\}}, \tag{2.6}$$

where $\mathrm{vol}(S)$ is the volume of the subset $S \subset V$ defined in Formula (2.1).

Now, the relation between the Cheeger's constant and the second smallest eigenvalue ξ_1 of the normalized Laplacian matrix follows from the Cheeger's inequality:

$$\frac{\Phi(G)^2}{2} \leq \xi_1 \leq 2\Phi(G). \tag{2.7}$$

The proof of Formula (2.7) can be found, for instance, in (71). It is proved for unweighted graphs without self-loops but the proof can be generalized to weighted graphs with self-loops (see Section 5 in (71)).

A related results to the Cheeger's inequality, is represented by he following formula, that is again shown in (49) :

$$\xi_1 \geq \frac{1}{D(G)\,\mathrm{vol}\,(G)}\,, \tag{2.8}$$

where $D(G)$ is the diameter of G, i.e., the maximum value of the shortest paths between any pair of nodes. From Formula (2.8), we can infer that $\mathrm{vol}\,(G)$ being the same, ξ_1 is larger when G is a "cluster" of vertices. For instance, if we consider two subgraphs with the same volume, the smaller $D(G)$ (i.e., the more a subgraph is "dense" or a cluster), the larger the lower bound on ξ_1; it is then expected that ξ_1 is larger. We will see in Chapter 4 that Formulas (2.7) and (2.8) are useful to determine an upper and lower bound for the second smallest eigenvalue of the normalized Laplacian matrix ξ_1 that is needed to estimate the rate of convergence to the consensus state. In particular, we will see that, in order to achieve a fast convergence to the consensus state, one should avoid a situation where a graph, or a subgraph is made of two clusters "poorly" connected by a small number of edges, since in this case, the upper bound on ξ_1 and thus the value of ξ_1 is expected to be small.

2.4 Summary

In this chapter we provided a brief background on graph theory. The basic definitions and notions were introduced focusing on the description of some features related to both single nodes and set of nodes that constitute subgraphs. The basic notations of spectral graph theory and its related quantities and properties were introduced too.

In the next chapter we will focus on the use of graphs in the study of dynamical systems. Our attention is channeled on the consensus problem discussed first in Chapter 3 and then evaluated from a different point of view in Chapter 4. In particular, to deal with this kind of problem, the spectral properties of graphs will be evaluated and studied in details.

Subsequently in Chapters 5 and 6 graphs will be exploited in a machine learning context. Again, the spectral properties of graphs will be considered to introduce an application in a classification problem of a semi–supervised learning technique, while graph–based features will be evaluated in another classification problem in order to extract additional and useful information from a dataset.

Part II

The consensus problem and its graph of interconnections

3

The consensus problem and its sparse variations

3.1 Introduction

The theory of complex systems deals with the study of the behavior of different types of systems made of agents that interact among each other; the study, for instance, of physical systems with particles that interact, social and economic networks, natural and ecological systems are typical examples (72). All of these situations are characterized by a large number of agents which interact among each other, without having, in general, a global knowledge about the structure of the system but interacting only locally with their neighbors (73; 74).

One particular study characterized by a network of agents that locally

This chapter is partly based on:

- G. Gnecco, R. Morisi, A. Bemporad, "Sparse solutions to the average consensus problem via various regularizations of the fastest mixing Markov-chain problem", IEEE Transactions on Network Science and Engineering, 2:97-111,2015
- G. Gnecco, R. Morisi, A. Bemporad, "Sparse solutions to the average consensus problem via l_1–norm regularization of the fastest mixing Markov-chain problem", Proceedings of the IEEE International Conference on Decision and Control (CDC), 2014:2228-2233

interact among each other is the so–called consensus problem (30; 75). It consists in investigating how complex systems, constituted by the interconnection of many units, behave with respect to the possibility of reaching consensus. The behavior of these systems is related on the dynamics of their units and on the topology of the connections between them. In particular, we are interested in studying the conditions under which a complex system has all its agents agree with the same opinion (the consensus state) and the time of convergence to that consensus. In this work we focus on the study of the rate of convergence to the consensus state paying particular attention to the the structure of the graph of interconnections with the aim of keeping it as sparse as possible. In fact, when dealing with large graphs with high cost of communication between the different nodes, it reveals extremely useful to find a solution able to guarantee the convergence to the consensus state as fast as possible reducing the number of interconnections, thus the communication costs, between the different agents. Starting from the formulation given in (48), the main contribution of the present work resides in the study from both a theoretical and practical point of view of sparse solutions of the consensus problem with satisfactory rate of convergence to the consensus state.

We first provide an introduction to the model of the problem, see e.g., Section 3.2; then, in Section 3.3 we introduce the Fastest Mixing Markov–Chain (FMMC) Problem, which is concerned with the study of the topology of the network with the aim of determining the fastest mixing Markov chain on the graph (48). We then provide an additional theoretical study and some variations of the FMMC Problem (see e.g., Section 3.4); and finally, in Section 3.5 we show numerical results.

3.2 Consensus problem - the model

Let $G = (V, E)$ be a network of n agents (i.e., $|V| = n$) which locally interact between each other. We aim at studying the dynamics of the units in order to have all the agents agree upon a common opinion. This

problem can be modeled by the following linear system:

$$X(t+1) = PX(t), \tag{3.1}$$

where $X(t) \in \mathbb{R}^n$ is the vector describing the opinions of the n agents at a generic time t, while P is a symmetric doubly stochastic matrix, (i.e., $P_{ij} \geq 0 \; \forall \; i, j = 1, \ldots, n, \; P\mathbf{1}_n = \mathbf{1}_n$ and $P = P^T$), where $\mathbf{1}_n$ is the vector of length n with each entry equal to 1. In particular, P can be seen as a matrix of transition probabilities of a finite–states Markov chain, with $P_{ii} \geq 0$ for all $i \in 1, \ldots, n$. If the undirected graph $G = (V, E)$, with $|V| = n$, associated to the dynamical system described by equation (3.1) is connected, it is well–known (see, e.g., (30)) that

$$x_i(t) \overset{t \to \infty}{\longrightarrow} \frac{1}{n} \mathbf{1}_n^T x(0), \; \forall i = 1, \ldots, n; \tag{3.2}$$

with $x(0)$ being the vector of the initial opinions of the agents. The quantity $\Sigma = \frac{1}{n} \mathbf{1}_n^T x(0)$, that is exactly the average of the initial opinions of the agents, is the consensus state of the system. We aim at studying the convergence rate to the consensus state Σ making it as fast as possible. In addition, we are interested in obtaining sparse solutions to the average consensus problem. This is motivated, e.g., in the case of high cost of communication associated with each edge. Note that we are limiting our study on the time–invariant consensus algorithm described by Formula (3.1) where the network of agents is described by a time–invariant undirected graph. Anyway, time–variant models (where the matrix of interconnection $P(t)$ is time dependent) and with a directed network of agents are also possible (76).

Before diving in the specific study of the consensus problem, we provide some information about the matrix of interconnections P and how its spectral properties are related to the convergence rate to the consensus state.

3.2.1 Spectral properties of matrix P

Let us consider the graph $G = (V, E)$ associated to the dynamical system studied; thus $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes which cor-

respond to the agents of the system. P is the transition probability matrix associated to G with eigenvalues $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{n-1}$ ordered in a non-increasing order. Each entry P_{ij} different from zero is associated to an edge e_{ij} between the vertex v_i and v_j of G. In particular, we require G to be connected, otherwise the consensus state cannot be reached. Under these assumptions, from Perron–Frobenius theorem (77) (which we report below), it follows that P has a positive (real) eigenvalue $\lambda_{\max} = \lambda_0$ with multiplicity equal to 1 and all the other eigenvalues satisfy $|\lambda_i| < \lambda_{\max}$ with $i = 1, \ldots, n-1$.

Theorem 1 (Perron-Frobenius theorem)**.** *Let $A \in \mathbb{R}^{n \times n}$ be an irreducible and non-negative matrix (i.e., the graph associated with the matrix is strongly connected), then:*

- *A has a positive (real) eigenvalue λ_{max} such that all the other eigenvalues of A satisfy*
$$|\lambda_i| < \lambda_{max}, \textit{with } i = 1, \ldots, n-1,$$
- *λ_{max} has algebraic and geometric multiplicity one, and has an eigenvector x with positive entries;*

In the following proposition we will use λ_0 instead of λ_{max}.

Proposition 1. *Let the assumptions of Theorem 1 holds, with $A = P$ a stochastic matrix. Then $\lambda_0 = 1$.*

Proof. From Perron Frobenius theorem we have that $\exists\ \lambda_0 > 0$ eigenvalue of P. Now, being v_0 the eigenvector associated to λ_0, we have that $Pv_0 = \lambda_0 v_0$. Considering the i-th component of the matrix–vector product Pv_0, this is equal to $(Pv_0)_i = \sum_j P_{ij} v_{0j} = (\lambda_0 v_0)_i = \lambda_0 v_{0i}$. Now, we define $M := \max_j(v_{0j})$, and $m := \min_j(v_{0j})$. It follows that

$$m = \sum_j P_{ij} m \leq (\lambda_0 v_0)_i \leq \sum_j P_{ij} M = M.$$

In particular, $\exists\ h$ such that $v_{0h} = m$, and $\exists\ k$ such that $v_{0k} = M$. Thus, we have:

$$m \leq \lambda_0 v_{0h} = \lambda_0 m \Rightarrow \lambda_0 \geq 1,$$
$$\lambda_0 M = \lambda_0 v_{0k} \leq M \Rightarrow \lambda_0 \leq 1.$$

Finally, from both the inequalities we can conclude that $\lambda_0 = 1$. □

From the previous statements we can now show that the rate of convergence to the consensus state is related to the second-largest eigenvalue modulus of P, $\mu(P) := \max_{j=1,\dots,n-1} |\lambda_j|$ (58; 78), with $\mu(P) = \lambda_1$. In fact, considering the Jordan decomposition of matrix P (79)

$$P = T^{-1} \left[\begin{array}{c|ccc} \lambda_0 & & & \\ \hline & \Lambda_1 & & \\ & & \ddots & \\ & & & \Lambda_m \end{array}\right] T,$$

where $T = \begin{bmatrix} w^T \\ \mathsf{W} \end{bmatrix}$, $T^{-1} = [v\ V] = [\mathbf{1}_n V]$ and $\Lambda := \mathrm{diag}\{\Lambda_1, \dots, \Lambda_m\}$ with Λ_i the Jordan blocks associated to the eigenvalues λ_i, it follows:

$$P = v\lambda_0 w^T + V\Lambda W = \mathbf{1}_n w^T + V\Lambda W,$$
$$P^t = \mathbf{1}_n w^T + V\Lambda^t W \overset{t\to\infty}{\longrightarrow} \mathbf{1}_n w^T.$$

Now, denoting with $c(t)$ the norm of the vector having as components the differences, at time t, between the state of each agent $x(t)$ and the consensus state $x(\infty)$, we have:

$$c(t) = \|x(t) - x(\infty)\| = \|(\mathbf{1}_n w^T + V\Lambda^t W)x(0) - \mathbf{1}_n w^T x(0)\| = \|V\Lambda^t W x(0)\|. \tag{3.3}$$

Thus, the rate of convergence to the consensus state is related to the largest eigenvalue of the matrix Λ and this is exactly λ_1, the second largest eigenvalue of P. Hence, the smaller $\lambda_1 = \mu(P)$ the faster the convergence rate to the consensus state. Since the symmetric matrix P has non–negative elements and satisfies $P\mathbf{1}_n = \mathbf{1}_n$, its generic element P_{ij} can be interpreted as a transition probability from the vertex i to the vertex j of a graph (including the case of a self-loop when $i = j$), whose vertices are the subsystems. Hence, the rate of convergence of the Markov chain with transition probabilities P_{ij} to its stationary distribution depends on $\mu(P)$.

A related quantity to $\mu(P)$ is the mixing time (48)

$$\tau(P) := \frac{1}{\log\left(\frac{1}{\mu(P)}\right)}, \tag{3.4}$$

which is an asymptotic measure of the number of steps required for reducing by the Euler's number e a suitable distance (the total variation distance) between the global state vector and the vector whose components are equal to the average consensus state.

Note that, from now on, we continue to indicate with $\mathbf{1}_s$ a generic s–dimensional vector with entries equal to 1, and analogously, with $\mathbf{0}_s$ we will indicate the s–dimensional vector with all the entries equal to 0.

3.3 The Fastest Mixing Markov Chain Problem

We are interested in increasing the rate of convergence to the consensus state, thus, given the propositions stated above and Formula (3.3), we aim at minimizing $\mu(P)$. The problem of determining the coefficients P_{ij} that minimize $\mu(P)$ subject to a given topology of the graph is called the Fastest Mixing Markov–Chain problem (Problem FMMC, in the following). In particular, we start from the formulation given in (48), where the problem of minimizing the value of $\mu(P)$ is treated as a convex optimization problem (specifically, as a semidefinite program (SDP)), and it is the following:

$$\textbf{Problem FMMC (first formulation)}:$$

$$\begin{array}{ll} \text{minimize}_{P\in\mathbb{R}^{n\times n}} & \mu(P) \\ \text{subject to} & P\mathbf{1}_n = \mathbf{1}_n,\ P = P^T, \\ & P_{ij} \geq 0,\ \forall i,j \in \{1,\ldots,n\}, \\ & P_{ij} = 0,\ \text{if } (i,j) \notin \mathrm{E}. \end{array} \tag{3.5}$$

Interestingly, this is a convex optimization problem, since

$$\mu(P) = |\lambda_{\max}|\left\{P - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\right\} \tag{3.6}$$

(see (48) for a proof of formula (3.6)), where $|\lambda_{\max}|$ stands for the largest eigenvalue modulus. Moreover, Problem FMMC can also be written as a semidefinite program (48, Section 2.3).

We first introduce an equivalent version of Problem FMMC, using a notation suitable for its sparse extensions presented later and for their theoretical investigations.

In the following, we denote by $w \in \mathbb{R}^m$ the column vector of weights associated with the m edges joining different vertices, and by $w_{\rm sl} \in \mathbb{R}^n$ the column vector of weights associated with the n self-loop edges. Hence, we can represent the weighted adjacency matrix P as a linear function $P(w, w_{\rm sl})$ of such weights. For instance, for $n = 3$ and $m = n(n-1)/2$ (the case of a complete graph), one obtains the symmetric matrix

$$P(w, w_{\rm sl}) = \begin{bmatrix} w_{{\rm sl},1} & w_1 & w_2 \\ w_1 & w_{{\rm sl},2} & w_3 \\ w_2 & w_3 & w_{{\rm sl},3} \end{bmatrix} . \tag{3.7}$$

Moreover, introducing the vertex-edge incidence matrix $M \in \mathbb{R}^{n \times m}$, whose elements are defined as follows:

$$M_{ij} = \begin{cases} 1\,, & \text{if the vertex } v_i \text{ is an endpoint of} \\ & \text{the (non self-loop) edge } e_{ij}\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{3.8}$$

and setting

$$w_{\rm sl} := \mathbf{1}_n - Mw\,, \tag{3.9}$$

the constraints

$$P_{ij} \geq 0 \text{ for any } i, j \in \{1, \ldots, n\} \text{ and } P\mathbf{1}_n = \mathbf{1}_n \tag{3.10}$$

are equivalent to

$$w_i \geq 0 \text{ for any } i \in \{1, \ldots, m\} \text{ and } Mw \leq \mathbf{1}_n\,. \tag{3.11}$$

Using (3.9), the matrix P becomes an affine function $P(w)$ of the weights vector w, and the second-largest eigenvalue modulus of P is expressed as a convex function - denoted by $\mu(w)$ - of the weights vector w, since convexity is preserved by affine mappings (80, Section 3.2). With the notations just introduced, Problem FMMC can be compactly rewritten as

Problem FMMC (second formulation) :

$$\begin{aligned} \text{minimize}_{w \in \mathbb{R}^m} \quad & f(w) := \mu(w) \\ \text{subject to} \quad & w \geq \mathbf{0}_m, \\ & Mw \leqslant \mathbf{1}_n \,. \end{aligned} \tag{3.12}$$

Again, we assume the graph associated to the vertex–edge incidence matrix M to be connected, in order to have $\lambda_0 = \lambda_{max}$ with multiplicity equal to 1.

We now start dealing with sparse variations of the problem just introduced.

3.4 Sparse variations of the Fastest Mixing Markov – Chain problem

Our study concerns the determination of a trade–off between good spectral properties of the communication graph involved in the consensus problem, hence of the transition probability matrix P, and its sparsity. To achieve this goal, we intend to use an approach based on an l_1-norm regularized version of Problem FMMC, which is called Problem FMMC-$l_1(\eta)$, where $\eta > 0$ denotes the regularization parameter. This variation of Problem FMMC is motivated by the fact that, due to geometrical properties of the l_1-norm (81), the introduction of such a regularization term in the objective of a convex optimization problem often enforces the sparsity of an optimal solution of the regularized version of that problem. In a second moment, we also consider another variation of Problem FMMC (called Problem FMMC$_{\text{constr}}$-$l_1(\eta)$) in which, besides the introduction of the l_1-norm regularization term, the weights of some edges are fixed. Notice that similar studies related to the determination of sparse solutions of the problem have been already carried on, for instance, in (82, Section 7.2) and in (83). In particular, Problems FMMC-$l_1(\eta)$ and FMMC$_{\text{constr}}$-$l_1(\eta)$ are similar to one already proposed and investigated numerically in (82, Section 7.2), with the difference that the l_1-norm term in that reference appears inside an additional constraint instead than in the objective. We also mention that, for the average consensus problem in the presence of disturbances, a similar

graph-sparsification optimization problem was also recently considered in (83), and solved through the Alternating Direction Method of Multipliers (ADMM) (84). Unlike these previous works, we also give an interpretation of Problem FMMC-$l_1(\eta)$ as a robust version of the Fastest Mixing Markov-Chain problem, and provide theoretical results using Gershgorin's theorem and Weyl's inequalities (79). Finally, we investigate the regularized version of Problem FMMC obtained by replacing the l_1-norm in Problem FMMC-$l_1(\eta)$ with the l_0-"pseudo-norm" $\|w\|_0 := \text{number of non-zero components of } w$. Both, a theoretical analysis of such variations of Problem FMMC and a numerical example modeling a wireless sensor network are provided, comparing their solutions with the one obtained solving Problem FMMC.

3.4.1 Sparse variations of Problem FMMC - models

We define for any $\eta > 0$, the following regularized version of Problem FMMC, in which an l_1-regularization term with regularization parameter η is added to the objective (here, $\|w\|_1 := \sum_{i=1}^{m} |w_i|$):

$$\textbf{Problem FMMC} - l_1(\eta):$$

$$\begin{array}{ll} \text{minimize}_{w \in \mathbb{R}^m} & f^{(1,\eta)}(w) := \mu(w) + \eta\|w\|_1 \\ \text{subject to} & w \geq \mathbf{0}_m, \\ & Mw \leqslant \mathbf{1}_n\,. \end{array} \tag{3.13}$$

The term $\eta\|w\|_1$ in (3.13) often induces sparsity of a resulting optimal solution $w^\circ(\eta)$ (81), i.e., many components of $w^\circ(\eta)$ tend to be 0.

In addition, we consider the regularized version of Problem FMMC considering the l_0-pseudo norm instead of the l_1-norm. This problem is defined as follows:

$$\textbf{Problem FMMC} - l_0(\eta):$$

$$\begin{array}{ll} \text{minimize}_{w \in \mathbb{R}^m} & f^{(0,\eta)}(w) := \mu(w) + \eta\|w\|_0 \\ \text{subject to} & w \geq \mathbf{0}_m, \\ & Mw \leqslant \mathbf{1}_n\,. \end{array} \tag{3.14}$$

Although the l_0-pseudo-norm is a more natural way to enforce sparsity than the l_1-norm, it is a nonconvex function, so, when sparsity is desired, it is common to replace the l_0-pseudo-norm with the l_1-norm, which is a convex function.

Another variation of Problem FMMC-$l_1(\eta)$ we intend to deal with, consists in fixing some components of the weight vector w. This is motivated, e.g., when one is interested in imposing some additional structure on the topology of the graph resulting from the optimization of the weight vector (e.g., enforcing the presence of given subgraphs, such as trees connecting important "backbone" vertices). Without loss of generality, we assume (up to a permutation of the indices) that the fixed weights are the first m_{fixed} ones (where $1 \leq m_{\text{fixed}} \leq m$), whereas the last $m_{\text{free}} : m - m_{\text{fixed}}$ weights are not fixed (the special case $m_{\text{fixed}} = m$ is trivial). We then decompose the column vector w as

$$w = \text{col}(w_{\text{fixed}}, w_{\text{free}}) \tag{3.15}$$

and the vertex-edge incidence matrix M as

$$M = [M_{\text{fixed}}|M_{\text{free}}]\,, \tag{3.16}$$

and we express the second-largest eigenvalue modulus μ as a function $\mu(w_{\text{free}})$ of the unfixed weights only. Then, for a given choice of the weight vector w_{fixed}, we consider the following optimization problem:

$$\textbf{Problem FMMC}_{\textbf{constr}} - l_1(\eta):$$

$$\begin{array}{ll} \text{minimize}_{w_{\text{free}} \in \mathbb{R}^{m_{\text{free}}}} & f^{(1,\eta)}_{\text{constr}}(w_{\text{free}}) := \mu(w_{\text{free}}) + \eta \|w_{\text{free}}\|_1 \\ \text{subject to} & w_{\text{free}} \geq \mathbf{0}_{m_{\text{free}}}, \\ & M_{\text{free}} w_{\text{free}} \leqslant \mathbf{1}_n - M_{\text{fixed}} w_{\text{fixed}}\,. \end{array} \tag{3.17}$$

Problem FMMC$_{\text{constr}}$-$l_1(\eta)$ has a form which is similar to the one of Problem FMMC-$l_1(\eta)$. We assume in the following that the fixed weights have been chosen in such a way that the polyhedron

$$\{w \in \mathbb{R}^{m_{\text{free}}} : w_{\text{free}} \geq \mathbf{0}_{\text{free}}, M_{\text{free}} w_{\text{free}} \leqslant \mathbf{1}_n - M_{\text{fixed}} w_{\text{fixed}}\} \tag{3.18}$$

is non-empty, so that Problem FMMC$_{\text{constr}}$-$l_1(\eta)$ admits a feasible solution.

Remark 3. *A similar variation can be studied replacing the l_1-norm with the l_0-pseudo-norm, but it is not investigated here, to avoid redundancy in the analysis. Another variation is obtained assuming that some edges are just "sufficiently used" rather than "fixed", i.e., that, for some indices i and some constants $\beta_i \in [0,1]$, one has $w_i \geq \beta_i$. The resulting problem has still linear equality and inequality constraints.*

Remark 4. *Although the l_1-norm is nondifferentiable at the origin, the terms $\|w\|_1$ and $\|w_{\text{free}}\|_1$ in the objectives of Problems FMMC$_{\text{constr}}$-$l_1(\eta)$ and FMMC-$l_1(\eta)$ can also be written, respectively, as $\mathbf{1}_m^T w$ and $\mathbf{1}_{m_{\text{free}}}^T w_{\text{free}}$ (thus, as linear - hence differentiable - terms), due to the respective non-negativity constraints $w \geq \mathbf{0}_m$ and $w_{\text{free}} \geq \mathbf{0}_{m_{\text{free}}}$.*

3.4.2 Theoretical results for Problems FMMC-$l_1(\eta)$ and FMMC$_{\text{constr}}$-$l_1(\eta)$

We first consider a theoretical analysis of Problem FMMC-$l_1(\eta)$, and then we extend the results to Problem FMMC$_{\text{constr}}$-$l_1(\eta)$. Note that, for a clearer notation, in the following definitions, propositions and remarks, when introducing a generic s–dimensional vector different from $w \in \mathbb{R}^m$, the s–dimensional zero vector $\mathbf{0}_s \in \mathbb{R}^s$ and the vector $\mathbf{1}_s \in \mathbb{R}^s$ with entries equal to 1, we will denote it with $\underline{h} \in \mathbb{R}^s$.

a) Existence of an optimal solution.

Proposition 2. Problem FMMC-$l_1(\eta)$ admits an optimal solution for every $\eta > 0$.

Proof. The feasible set of Problem FMMC-$l_1(\eta)$ is convex, closed, and bounded. Moreover, its objective is continuous since the l_1-norm regularization term $\|w\|_1$ is continuous, and on the feasible set the second-largest eigenvalue modulus $\mu(P)$ has the expression (3.6), which is continuous due to the continuous dependence of the eigenvalues of a matrix on its entries (85, Section 7.6), and the fact that the point-wise maximum of a finite set of continuous functions is continuous, too. Concluding, Problem FMMC-$l_1(\eta)$ involves the minimization of a continuous objective function on a compact set, so an optimal solution to Problem FMMC-$l_1(\eta)$ exists by Weierstrass theorem. □

b) Effect of the regularization parameter.

Solving Problem FMMC-$l_1(\eta)$ involves finding a good compromise between the minimization of the term $\mu(w)$ and the one of $\|w\|_1$. Next Proposition 3 shows that the regularization parameter η has opposite effects on the two terms $\mu(w)$ and $\|w\|_1$, when evaluated at an optimal solution.

Proposition 3. Let $0 < \eta_1 < \eta_2$, and $w_1^\circ(\eta_1), w_1^\circ(\eta_2)$ be optimal solutions to Problem FMMC-$l_1(\eta_1)$ and Problem FMMC-$l_1(\eta_2)$, respectively. Then,
i) $\mu(w_1^\circ(\eta_1)) \leq \mu(w_1^\circ(\eta_2))$,
ii) $\|w_1^\circ(\eta_1)\|_1 \geq \|w_1^\circ(\eta_2)\|_1$.

Proof. By the optimality of $w_1^\circ(\eta_1)$ for Problem FMMC-$l_1(\eta_1)$, one has

$$\mu(w_1^\circ(\eta_1)) + \eta_1\|w_1^\circ(\eta_1)\|_1 \leq \mu(w_1^\circ(\eta_2)) + \eta_1\|w_1^\circ(\eta_2)\|_1 \,. \tag{3.19}$$

Similarly, by the optimality of $w_1^\circ(\eta_2)$ for Problem FMMC-$l_1(\eta_2)$, one gets

$$\mu(w_1^\circ(\eta_2)) + \eta_2\|w_1^\circ(\eta_2)\|_1 \leq \mu(w_1^\circ(\eta_1)) + \eta_2\|w_1^\circ(\eta_1)\|_1 \,. \tag{3.20}$$

Combining the two inequalities above, one obtains

$$\begin{aligned} &\eta_2(\|w_1^\circ(\eta_2)\|_1 - \|w_1^\circ(\eta_1)\|_1) \\ \leq &\mu(w_1^\circ(\eta_1)) - \mu(w_1^\circ(\eta_2)) \\ \leq &\eta_1(\|w_1^\circ(\eta_2)\|_1 - \|w_1^\circ(\eta_1)\|_1) \,, \end{aligned}$$

which is satisfied if and only if conditions i) and ii) hold, as $0 < \eta_1 < \eta_2$. □

In general, instead, the sparsity

$$\begin{aligned} s(w_1^\circ(\eta)) &:= 1 - \|w_1^\circ(\eta)\|_0/m \\ &= \text{ fraction of zero components of } w^\circ(\eta) \end{aligned}$$

of an optimal solution to Problem FMMC-$l_1(\eta)$ may not be a monotonic function of η, as shown in Fig. 5 (see Section 3.5). This behavior is similar to the one observed for other l_1-regularized optimization problems, such as the classical Least Absolute Shrinkage and Selection Operator (LASSO) problem (see., e.g., Fig. 1 in (86)).

c) Conditions under which $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_1(\eta)$.

The next result states conditions on the regularization parameter under which $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_1(\eta)$, or its unique optimal solution. An application of the result to the choice of the regularization parameter is given in Section 3.4.2 e).

Proposition 4. Let $\eta \geq 2$. Then $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_1(\eta)$. If $\eta > 2$, then $w = \mathbf{0}_m$ is its unique optimal solution.

Proof. Let Δw be an arbitrary admissible increment of w, starting from $w = \mathbf{0}_m$ (such an increment can be used to generate the whole set of admissible solutions to Problem FMMC-$l_1(\eta)$, since it is convex and contains $\mathbf{0}_m$). Then, the corresponding increment $\Delta f^{(1,\eta)}$ in the objective $f^{(1,\eta)}$ of Problem FMMC-$l_1(\eta)$ is

$$\Delta f^{(1,\eta)} = \mu(\Delta w) - \mu(\mathbf{0}_m) + \eta \|\Delta w\|_1 \,. \tag{3.21}$$

Now, one has $\mu(\mathbf{0}_m) = 1$ (as the associated weighted adjacency matrix is $P(\mathbf{0}_m) = I_n$, the identity matrix of dimension $n \times n$), whereas one can find a lower bound on $\mu(\Delta w)$ as follows. The matrix $P(\Delta w)$ can be written as

$$P(\Delta w) = I_n + E(\Delta w) \,, \tag{3.22}$$

where the main-diagonal entries of the matrix $E(\Delta w)$ are non-positive with their absolute values bounded from above by $\|\Delta w\|_1$, whereas, for each row i, one has

$$\sum_{j \neq i,\ j=1}^{n} |E_{ij}(\Delta w)| \leq \|\Delta w\|_1 \,. \tag{3.23}$$

Then, by Gersghorin's theorem [1], all the eigenvalues of $E(\Delta w)$ are bounded from above in absolute value by $2\|\Delta w\|_1$. As the presence of the matrix I_n in formula (3.22) has only the effect of translating the eigenvalues of $E(\Delta w)$ by 1, one finally obtains

$$\mu(\Delta w) \geq 1 - 2\|\Delta w\|_1 \,. \tag{3.24}$$

Concluding, for an arbitrary admissible increment Δw, one gets

$$\Delta f^{(1,\eta)} \geq 1 - 2\|\Delta w\|_1 - 1 + \eta \|\Delta w\|_1 = -2\|\Delta w\|_1 + \eta \|\Delta w\|_1 \,. \tag{3.25}$$

[1] Gersghorin's theorem (87, Section 7.2) states that all the eigenvalues of a matrix $A \in \mathbb{R}^{n \times n}$ belong, in the complex plane, to at least one of the Gersghorin circles G_i (for $i = 1, \ldots, n$), whose centers and radii are defined, respectively, by A_{ii} and $\sum_{j \neq i,\ j=1}^{n} |A_{ij}|$.

If $\eta \geq 2$, then $\Delta f^{(1,\eta)}$ is non-negative for every admissible Δw, hence $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_1(\eta)$. If $\eta > 2$, then $\Delta f^{(1,\eta)}$ is positive for every $\Delta w \neq \mathbf{0}_m$, hence $w = \mathbf{0}_m$ is the only optimal solution to Problem FMMC-$l_1(\eta)$. □

The following example shows that the bound obtained in Proposition 4 is tight, at least if one does not impose further restrictions on the class of graphs to be considered.

Example 1. *Let $n = 2$ and $m = 1$. Then, the matrix $P(w)$ has the expression*

$$P(w) = \begin{bmatrix} 1 - w_1 & w_1 \\ w_1 & 1 - w_1 \end{bmatrix}, \tag{3.26}$$

whose eigenvalues are 1 and $1 - 2w_1$. Hence, on the set $[0, 1]$ of admissible solutions to Problem FMMC-$l_1(\eta)$, its objective is $|1-2w_1|+\eta w_1$, and $w_1 = 0$ is, respectively, the unique optimal solution to Problem FMMC-$l_1(\eta)$ for $\eta > 2$, one of its (infinite) optimal solutions for $\eta = 2$ (the optimal ones being all $w_1 \in \left[0, \frac{1}{2}\right]$), and a suboptimal solution for $0 < \eta < 2$ (the optimal one being $w_1 = \frac{1}{2}$).

d) Non differentiability of the objective at $w = \mathbf{0}_m$.

In the proof of Proposition 4, we have used Gersghorin's theorem instead than an approach based on first-order optimality conditions for a differentiable objective. The reason is that, as shown in the next Proposition 5, in general the objective of Problem FMMC-$l_1(\eta)$ is not differentiable at $w = \mathbf{0}_m$, although the l_1-regularization term is represented by a linear function on its set of admissible solutions. To state Proposition 5, we need the following definition.

Definition 1. A function $y : D \to \mathbb{R}$ defined on a convex and compact subset $D \subseteq \mathbb{R}^m$ is differentiable at a point $\underline{x}_0$ belonging to the boundary ∂D of D iff there exists a linear map $J : \mathbb{R}^m \to \mathbb{R}$ such that

$$\lim_{t \to 0^+} \frac{y(\underline{x}_0 + t\underline{h}) - y(\underline{x}_0) - J(t\underline{h})}{\|t\underline{h}\|} = 0\,, \tag{3.27}$$

for all feasible directions $\underline{h} \in \mathbb{R}^m$, i.e., such that $\underline{x}_0 + t\underline{h} \in D$ for all $t > 0$ sufficiently small.

Remark 5. *In the usual definition of differentiability, instead, one assumes that the point $\underline{x}_0$ belongs to the interior of the set D, and in that case all vectors $\underline{h} \in \mathbb{R}^m$ are feasible directions. In Definition 1, we extend this concept to a point belonging to the boundary of the domain (which is the case of $w = \mathbf{0}_m$ for Problem FMMC-$l_1(\eta)$, as it belongs to the boundary of the domain of the function $\mu(w)$).*

Remark 6. *For $i = 1, \ldots, n$, let $\underline{e}_i \in \mathbb{R}^m$ denote the vector whose component i is equal to 1, and all its other components are equal to 0. If the directions $\underline{e}_i$ are feasible, (3.27) and the linearity of the map J imply, for every feasible direction $\underline{h}$,*

$$\lim_{t \to 0^+} \frac{y(\underline{x}_0 + t\underline{h}) - y(\underline{x}_0)}{\|t\underline{h}\|} = \sum_{i=1}^{m} h_i \lim_{t \to 0^+} \frac{y(\underline{x}_0 + t\underline{e}_i) - y(\underline{x}_0)}{t}.$$

Proposition 5. Let $n \geq 3$. Then the objective of Problem FMMC-$l_1(\eta)$ is nondifferentiable at $w = \mathbf{0}_m$.

Proof. For $i = 1, \ldots, n$ and for each $t \in [0, 1]$, the graph associated with the weighted adjacency matrix $P(t\underline{e}_i)$ is disconnected, hence the maximum (and maximum modulus) eigenvalue $\lambda_{\max}\{P(t\underline{e}_i)\} = 1$ of $P(t\underline{e}_i)$ has multiplicity at least 2, and its second-largest eigenvalue modulus is $\mu(t\underline{e}_i) = 1$. Thus, at $w = \mathbf{0}_m$, the directional derivative of the objective of Problem FMMC-$l_1(\eta)$ in the direction $\underline{e}_i$, i.e.,

$$\lim_{t \to 0^+} \frac{\mu(t\underline{e}_i) - \mu(\mathbf{0}_m) + \eta \mathbf{1}_m^T (t\underline{e}_i)}{t}, \tag{3.28}$$

is equal to η. Now, let $\hat{w}$ be any admissible weight vector with $\hat{w} > \mathbf{0}_m$ (elementwise). Since the graph associated with the weighted adjacency matrix $P(\hat{w})$ is connected, $\lambda_{\max}\{P(\hat{w})\} = 1$ has multiplicity equal to 1, hence $\lambda_2(P(\hat{w})) \leq \mu(\hat{w}) < 1$, and $\lambda_n\{P(\hat{w})\} > -1$ by Gersghorin's theorem. Now, for any $t \in [0, 1]$, one has $P(t\hat{w}) = (1 - t)I + tP(\hat{w})$. Then, the eigenvalues of $P(\hat{w})$ and $P(t\hat{w})$, ordered nonincreasingly with their multiplicity, are related by

$$\lambda_i\{P(t\hat{w})\} = (1 - t) + t\lambda_i\{P(\hat{w})\}, \quad (i = 1, \ldots, n), \tag{3.29}$$

so, assuming without loss of generality $\mu(\hat{w}) = \lambda_2(P(\hat{w})) > 0$ (the case $mu(\hat{w}) = \max\{|\lambda_2(P(\hat{w}))|, |\lambda_n(P(\hat{w}))|\}$ is similar), for $t > 0$ sufficiently

small one gets $\mu(t\hat{w}) = 1 - t + t\lambda_2(P(\hat{w}))$. Hence, at $w = \mathbf{0}_m$, the directional derivative of the objective of Problem FMMC-$l_1(\eta)$ in the direction $\hat{w}$, i.e.,

$$\lim_{t \to 0^+} \frac{\mu(t\hat{w}) - \mu(\mathbf{0}_m) + \eta \mathbf{1}_m^T (t\hat{w})}{t}, \tag{3.30}$$

is equal to $\lambda_2(P(\hat{w})) - 1 + \eta \|\hat{w}\|_1$, which differs from

$$\sum_{i=1}^{m} \hat{w}_i \left(\lim_{t \to 0^+} \frac{\mu(t\underline{e}_i) - \mu(\mathbf{0}_m) + \eta \mathbf{1}_m^T (t\underline{e}_i)}{t} \right) = \eta \|\hat{w}\|_1 \tag{3.31}$$

(due to Definition 1 and Remark 6, formulas (3.30) and (3.31) would have coincided, instead, in the case of differentiability of the objective of Problem FMMC-$l_1(\eta)$ at $w = \mathbf{0}_m$). Then, we conclude that the objective of Problem FMMC-$l_1(\eta)$ is nondifferentiable at $w = \mathbf{0}_m$. □

Remark 7. *For $n = 2$ and $m = 1$, instead, the objective of Problem FMMC-$l_1(\eta)$ is differentiable at $w = \mathbf{0}_m$, as shown by Example 1. This is not in contrast with the proof of Proposition 5, since in this case - as being $n < 3$ - the graph associated with the weighted adjacency matrix $P(t\underline{e}_1)$ is connected for every $t \in (0, 1]$.*

e) Choice of the regularization parameter and reoptimization.

The theoretical results presented above justify the following practical rule for choosing the regularization parameter η:

- given a positive integer N and a maximal acceptable increase $\varepsilon > 0$ for the second-largest eigenvalue modulus of P with respect to its optimal value μ°_{FMMC} in Problem FMMC, solve Problem FMMC-$l_1(\eta)$ in correspondence of N values $\eta^{(j)}$ for η such that
  - $0 < \eta^{(1)} < \eta^{(2)} < \ldots < \eta^{(N)} < 2$ (the last inequality needed to avoid just the trivial optimal solution $w^\circ = \mathbf{0}_m$), and
  - $\mu(w_1^\circ(\eta^{(j)})) \leq \mu^\circ_{FMMC} + \varepsilon$ $(j = 1, \ldots, N)$ (the inequality needed just to guarantee the desired tolerance on $\mu(w_1^\circ(\eta^{(j)}))$, hence on the rate of convergence to the consensus state);
- choose $j^\circ \in \{1, \ldots, N\}$ that maximizes the sparsity $s(w_1^\circ(\eta^{(j)}))$;

– perform a final reoptimization step, solving Problem FMMC on the graph obtained removing from the original graph all the non self-loop edges i for which $w^\circ_{1,i}(\eta^{(j^\circ)}) = 0$, obtaining another weight vector $w^\circ_{\rm reopt}$.

The last reoptimization step satisfies $\mu(w^\circ_{\rm reopt}) \leq \mu(w^\circ_1(\eta^{(j^\circ)}))$ (due to the optimality of $w^\circ_{\rm reopt}$ on Problem FMMC on the new graph, and the feasibility of $w^\circ_1(\eta^{(j^\circ)})$ for such an optimization problem), and $s(w^\circ_{\rm reopt}) \geq s(w^\circ_1(\eta^{(j^\circ)}))$. Such a reoptimization step is common to other l_1-regularized optimization problems: for the LASSO, it is known as debiasing (88, Section 13.3.5).

Finally, a possible way to choose the tolerance parameter ε (which has to be in any case smaller than $1 - \mu^\circ_{FMMC}$, again to avoid trivial optimal solutions) consists in expressing it in terms of the maximal allowable ratio ρ between the mixing time $\tau(P(w))$ and its optimal value $\tau^\circ_{FMMC} := \frac{1}{\log\left(\frac{1}{\mu^\circ_{FMMC}}\right)}$, which is obtained when solving Problem FMMC, i.e., one sets

$$\varepsilon = (\mu^\circ_{FMMC})^{\frac{1}{\rho}} - \mu^\circ_{FMMC}\,. \tag{3.32}$$

f) Interpretation of Problem FMMC-$l_1(\eta)$ as a robust version of Problem FMMC.

Problem FMMC-$l_1(\eta)$ has also the following interpretation. Let us suppose that, for any given "nominal" choice of the weights w_i ($i = 1, \ldots, m$), one has an "uncertainty" Δw_i such that $|\Delta w_i| \leq \delta |w_i|$, for some fixed $\delta > 0$. Then, an application of Gersghorin's theorem and Weyl's inequalities[2] in matrix-perturbation theory shows that the second-largest eigenvalue modulus $\mu(w + \Delta w)$ is bounded from above as

$$\mu(w + \Delta w) \leq \mu(w) + 2\delta \|w\|_1\,. \tag{3.36}$$

[2]Let $A, B \in \mathbb{R}^{n\times n}$ and symmetric, and let their eigenvalues be ordered nonincreasingly with their multiplicity as

$$\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_j(A) \geq \ldots \geq \lambda_n(A)\,, \tag{3.33}$$

$$\lambda_1(B) \geq \lambda_2(B) \geq \ldots \geq \lambda_j(B) \geq \ldots \geq \lambda_n(B)\,. \tag{3.34}$$

Then, in their simplest form, Weyl's inequalities (89, Theorem 8.4.11) state that, for every $j = 1, \ldots, n$, one has

$$\lambda_j(A) + \lambda_n(B) \leq \lambda_j(A + B) \leq \lambda_j(A) + \lambda_1(B)\,. \tag{3.35}$$

Then, an optimal "robust" choice of the nominal weight vector w is obtained minimizing the objective $\mu(w) + 2\delta\|w\|_1$ on the set of admissible weight vectors w, i.e., solving a robust version of Problem FMMC which takes into account the uncertainty of the weights, and is obtained replacing its objective $\mu(w)$ with $\mu(w) + 2\delta\|w\|_1$. However, this is equivalent to solving Problem FMMC-$l_1((\eta)$ with the choice $\eta = 2\delta$. Finally, we notice that, when $\delta \geq 1$, for every admissible nominal choice of the vector w, the perturbation $\Delta w = -w$ is admissible for the robust version of Problem FMMC just described, and the resulting perturbed vector is $w + \Delta w = \mathbf{0}_m$, which satisfies $\mu(\mathbf{0}_m) = 1$, as being the resulting graph disconnected. Hence, when $\delta \geq 1$, for every nominal choice of w one cannot have $\mu(w + \Delta w) < 1$ for every admissible perturbation, and $w = \mathbf{0}_m$ is an optimal nominal choice. This is consistent with Proposition 4.

g) Extension to Problem FMMC$_{\text{constr}}$-$l_1(\eta)$.

Apart from the tightness of the bound on the minimal value of the regularization parameter η for which $w_{\text{free}} = \mathbf{0}_{m_{\text{free}}}$ is an optimal solution, the results above can be extended to Problem FMMC$_{\text{constr}}$-$l_1(\eta)$. In particular, Propositions 2, 3, and 4 can be extended to Problem FMMC$_{\text{constr}}$-$l_1(\eta)$, simply replacing w_{free} with w. For the first two propositions, the extension requires no significant changes in the proofs. In the third case, the only significant change in the proof is the additional use of the abovementioned Weyl's inequalities to get a formula similar to (3.24), bounding the eigenvalues of the sum of two symmetric matrices. To obtain the extension of Proposition 5, maintaining the structure of the proof, one requires the additional assumption that the subgraph containing only the fixed edges is disconnected, and remains disconnected when adding arbitrarily only one of the "free" edges.

h) Formulation through semidefinite programming (SDP).

Likewise Problem FMMC, Problems FMMC-$l_1(\eta)$ and FMMC$_{\text{constr}}$-$l_1(\eta)$ can be formulated as semidefinite programs, allowing the use of interior-point methods for finding their optimal solutions. More precisely, expressing the m non self-loop edges in terms of their endpoints

as (i,j), and considering the set

$$\begin{aligned}\mathcal{E} :=& \{(i,j) : i \neq j,\ i,j \in \{1,\dots,n\},\ \text{and} \\ & \exists k \in \{1,\dots,n\} \text{ such that } M_{ik} = M_{jk} = 1\}. \end{aligned} \tag{3.37}$$

one obtains the following alternative formulation[3] of Problem FMMC-$l_1(\eta)$,

$$\textbf{Problem FMMC} - l_1(\eta)\ (\textbf{SDP formulation}) :$$

$$\begin{aligned} \text{minimize}_{s\in\mathbb{R}, P\in\mathbb{R}^{n\times n}} \quad & \left(s + \frac{\eta}{2} \sum_{i\neq j,\ i,j=1}^{n} P_{ij} \right) \\ \text{subject to} \quad & -sI \preceq P - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T \preceq sI, \\ & P\mathbf{1}_n = \mathbf{1}_n,\ P = P^T, \\ & P_{ij} \geq 0,\ \forall i,j \in \{1,\dots,n\}, \\ & P_{ij} = 0,\ \text{if } (i,j) \notin \mathcal{E}. \end{aligned} \tag{3.38}$$

Similarly to the previous expression, one gets the following alternative formulation of Problem FMMC$_{\text{constr}}$-$l_1(\eta)$, introducing the subset $\mathcal{E}_{\text{fixed}} \subseteq \mathcal{E}$ of edges (i,j) associated with fixed weights $P_{ij,\text{fixed}} = P_{ji,\text{fixed}}$.

$$\textbf{Problem FMMC}_{\textbf{constr}} - l_1(\eta)\ (\textbf{SDP formulation}) :$$

$$\begin{aligned} \text{minimize}_{s\in\mathbb{R}, P\in\mathbb{R}^{n\times n}} \quad & \left(s + \frac{\eta}{2} \sum_{i\neq j,\ i,j=1}^{n} P_{ij} \right) \\ \text{subject to} \quad & -sI \preceq P - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T \preceq sI, \\ & P\mathbf{1}_n = \mathbf{1}_n,\ P = P^T, \\ & P_{ij} \geq 0,\ \forall i,j \in \{1,\dots,n\}, \\ & P_{ij} = 0,\ \text{if } (i,j) \notin \mathcal{E}, \\ & P_{ij} = P_{ij,\text{fixed}},\ \text{if } (i,j) \in \mathcal{E}_{\text{fixed}}. \end{aligned} \tag{3.39}$$

[3] Here and in the following, for any two symmetric matrices $X, Y \in \mathbb{R}^{n\times n}$, the notation $X \preceq Y$ (resp., $X \succeq Y$) means that $X - Y$ is negative (resp., positive) semidefinite, whereas $X \leq Y$ (resp., $X \geq Y$) stands for an elementwise inequality.

Of course, the fixed weights P_{ij} can be removed from the summation inside the objective of the optimization problem above, without changing its optimal solution.

In Section 3.5, we present some numerical results obtained solving both Problems FMMC-$l_1(\eta)$ and FMMC$_{\text{constr}}$-$l_1(\eta)$ through a modified version of the MATLAB function `fmmc.m` in the CVX package (`http://cvxr.com/cvx/download/`), which solves the SDP formulation of Problem FMMC presented in (48) and (82). Such a modified version is basically obtained adding the linear term $\frac{\eta}{2}\sum_{i\neq j,\ i,j=1}^{n} P_{ij}$ inside the objective of the original optimization problem.

3.4.3 Theoretical results for Problem FMMC-$l_0(\eta)$

In this section, we provide some theoretical results about the optimal solution of Problem FMMC-$l_0(\eta)$. They usually provide structural properties of the optimal solution similar to the ones obtained in Section 3.4.2 for Problem FMMC-$l_1(\eta)$, with some differences in the proofs. To differentiate the notation with respect to the one used for Problem FMMC-$l_1(\eta)$, we use the subscript "0" when referring to an optimal solution of Problem FMMC-$l_0(\eta)$.

a) Existence of an optimal solution.

The next result states the existence of an optimal solution to Problem FMMC-$l_0(\eta)$. Before stating it, we need to recall the following definition of lower semi-continuity, which is used in the proof of the next Proposition 6.

Definition 2. Let $f : X \to \mathbb{R}$ be a real-valued function defined on a topological space X. Then, f is lower semi-continuous iff $\forall c \in \mathbb{R}$, the set $\{\underline{x} \in X : f(\underline{x}) \leq c\}$ is closed.

Proposition 6. Problem FMMC-$l_0(\eta)$ admits an optimal solution for every $\eta > 0$.

Proof. The feasible set of Problem FMMC-$l_0(\eta)$ is convex, closed, and bounded, hence compact, as being $\mathbb{R}^m$ a finite-dimensional vector space. Moreover, its objective is lower semi-continuous since:

– taking, for any $c > 0$, any convergent sequence $\{w^{(k)}\}_{k=1}^{\infty}$ such that $\|w^{(k)}\|_0 \leq c$ for all k, and observing that also its limit satisfies the same inequality, it follows that the l_0-pseudo-norm regularization term $\|w\|_0$ is lower semi-continuous;

– as it has been shown in the proof of Proposition 2, on the feasible set, the second-largest eigenvalue modulus $\mu(P(w))$ is continuous (hence, also lower semi-continuous);

– the sum of two lower semi-continuous functions is lower semi-continuous, too.

Concluding, Problem FMMC-$l_0(\eta)$ involves the minimization of a lower semi-continuous objective function on a compact set, so an optimal solution to Problem FMMC-$l_0(\eta)$ exists by the generalized Weierstrass theorem (90, Theorem 2.43). □

b) Effect of the regularization parameter.

Likewise Proposition 3, next Proposition 7 shows that the regularization parameter η has opposite effects on the two terms $\mu(w)$ and $\|w\|_0$, when evaluated at an optimal solution. The proof is very similar to the one of Proposition 3, hence it is not reported.

Proposition 7. Let $0 < \eta_1 < \eta_2$, and $w_0^\circ(\eta_1), w_0^\circ(\eta_2)$ be optimal solutions to Problem FMMC-$l_0(\eta_1)$ and Problem FMMC-$l_0(\eta_2)$, respectively. Then,
i) $\mu(w_0^\circ(\eta_1)) \leq \mu(w_0^\circ(\eta_2))$;
ii) $\|w_0^\circ(\eta_1)\|_0 \geq \|w_0^\circ(\eta_2)\|_0$.

Differently from the case of Problem FMMC-$l_1(\eta)$ investigated in Section 3.4.2, however, also the sparsity

$$\begin{aligned} s(w_0^\circ(\eta)) &:= 1 - \|w_0^\circ(\eta)\|_0/m \\ &= \text{ fraction of zero components of } w_0^\circ(\eta) \end{aligned}$$

is a monotonic function of η, as it is shown by the next proposition, whose proof is immediate and is, therefore, omitted.

Proposition 8. Let $0 < \eta_1 < \eta_2$, then $s(w_0^\circ(\eta_1)) \leq s(w_0^\circ(\eta_2))$.

Notice that, since

– any subgraph with n vertices and $\hat{m} \leq m$ (non self-loop) non-zero weighted edges cannot be connected when $\hat{m} < n-1$ (as $n-1$ is the number of edges in a spanning tree);

– when the subgraph associated with a feasible w is disconnected, the corresponding $\mu(w)$ is equal to 1, so the optimal choice for w in Problem FMMC-$l_0(\eta)$ is $w = \mathbf{0}_m$, when one limits to consider disconnected subgraphs;

the value of the optimal sparsity for Problem FMMC-$l_0(\eta)$ cannot belong to the interval $\left(\frac{m-n+1}{m}, 1\right)$ (which corresponds with disconnected subgraphs with at least 1 non self-loop edge), whereas the value 1 is achievable, and it corresponds to the trivial case of a completely disconnected subgraph at optimality. Concluding, the possible values for the optimal sparsity for Problem FMMC-$l_0(\eta)$ are $0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-n}{m}, \frac{m-n+1}{m}, 1$.

The next result states the continuity of the optimal value of the objective in Problem FMMC-$l_0(\eta)$, and shows also that Problem FMMC-$l_0(\eta)$ "behaves" like Problem FMMC for η sufficiently small.

Proposition 9. Let $w_0^\circ(\eta)$ (resp., w°) be an optimal solution of Problem FMMC-$l_0(\eta)$ (resp., of Problem FMMC), and $\mu(w_0^\circ(\eta))$ (resp., $\mu(w^\circ)$) the value of the second-largest eigenvalue modulus of the corresponding weighted adjacency matrix $P(w_0^\circ(\eta))$ (resp., $P(w^\circ)$). Then,
i) $\mu(w_0^\circ(\eta)) + \eta\|w_0^\circ(\eta)\|_0$ depends continuously on η;
ii) $\lim_{\eta\to 0^+} \mu(w_0^\circ(\eta)) = \mu(w^\circ)$;
iii) Given any sequence $\{\eta_k\}_{k=1}^{+\infty}$ convergent to 0, and an associated sequence $\{w_0^\circ(\eta_k)\}_{k=1}^{+\infty}$, one can extract from the latter a subsequence that converges to an optimal solution of Problem FMMC.

Proof. i) Let $0 < \eta_1$, $0 < \eta_2$, and $\eta_1 \neq \eta_2$. Due to the optimality of $w_0^\circ(\eta_1)$ and $w_0^\circ(\eta_2)$ for Problems FMMC-$l_0(\eta_1)$ and FMMC-$l_0(\eta_2)$, resp., one gets

$$\mu(w_0^\circ(\eta_1)) + \eta_1\|w_0^\circ(\eta_1)\|_0 \leq \mu(w_0^\circ(\eta_2)) + \eta_1\|w_0^\circ(\eta_2)\|_0\,, \tag{3.40}$$

and

$$\mu(w_0^\circ(\eta_2)) + \eta_2\|w_0^\circ(\eta_2)\|_0 \leq \mu(w_0^\circ(\eta_1)) + \eta_2\|w_0^\circ(\eta_1)\|_0\,. \tag{3.41}$$

Then, combining (3.40) and (3.41), one obtains

$$\begin{aligned}
&\mu(w_0^\circ(\eta_1)) \\
&\leq \mu(w_0^\circ(\eta_2)) + \eta_1 \left(\|w_0^\circ(\eta_2)\|_0 - \|w_0^\circ(\eta_1)\|_0\right) \\
&\leq \mu(w_0^\circ(\eta_1)) + \eta_2 \left(\|w_0^\circ(\eta_1)\|_0 - \|w_0^\circ(\eta_2)\|_0\right) \\
&\qquad + \eta_1 \left(\|w_0^\circ(\eta_2)\|_0 - \|w_0^\circ(\eta_1)\|_0\right) \\
&= \mu(w_0^\circ(\eta_1)) + (\eta_2 - \eta_1) \left(\|w_0^\circ(\eta_1)\|_0 - \|w_0^\circ(\eta_2)\|_0\right). \qquad (3.42)
\end{aligned}$$

Since $w_0^\circ(\eta_1)$ and $w_0^\circ(\eta_2)$ belong to the admissible set, which is compact, formula (3.42) implies (reversing also the roles of η_1 and η_2)

$$\begin{aligned}
&\mu(w_0^\circ(\eta_1)) + \eta_1 \|w_0^\circ(\eta_1)\|_0 \\
&= \lim_{\eta_2 \to \eta_1} \left(\mu(w_0^\circ(\eta_2)) + \eta_1 \|w_0^\circ(\eta_2)\|_0\right) \\
&= \lim_{\eta_2 \to \eta_1} \left(\mu(w_0^\circ(\eta_2)) + \eta_2 \|w_0^\circ(\eta_2)\|_0\right), \qquad (3.43)
\end{aligned}$$

from which item i) follows.

Item ii) is proved likewise item i), exploiting also the fact that w_0° belongs to an admissible compact set, hence the term $\eta \|w_0^\circ\|_0$ vanishes as η tends to 0 from the right.

Finally, item iii) is obtained combining item ii) with the compactness of the admissible set (which makes it possible to extract a convergent subsequence, starting from any subsequence belonging to that set) and the continuity of the second-largest eigenvalue modulus $\mu(P(w))$ with respect to w (already shown in the proof of Proposition 2). □

Remark 8. *A result similar to Proposition 9 holds also for Problem FMMC-$l_1(\eta)$. In that case, the proof of the corresponding item i) could be also obtained exploiting Berge's theorem of the maximum (see, e.g., (91, Section 3.3)).*

c) Conditions under which $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_0(\eta)$.

The next result states conditions on the regularization parameter under which $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_0(\eta)$, or its unique optimal solution. An application of the result to the choice of the regularization parameter is given in Section 3.4.3 f).

Proposition 10. Let $\eta \geq \frac{1}{n-1}$. Then $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_0(\eta)$. If $\eta > \frac{1}{n-1}$, then $w = \mathbf{0}_m$ is its unique optimal solution.

Proof. Let Δw be an arbitrary admissible increment of w, starting from $w = \mathbf{0}_m$. Then, the corresponding increment $\Delta f^{(0,\eta)}$ in the objective $f^{(0,\eta)}$ of Problem FMMC-$l_0(\eta)$ is

$$\Delta f^{(0,\eta)} = \mu(\Delta w) - \mu(\mathbf{0}_m) + \eta \|\Delta w\|_0 \,. \tag{3.44}$$

Now, one has $\mu(\mathbf{0}_m) = 1$ (as the associated weighted adjacency matrix is $P(\mathbf{0}_m) = I_n$). If $\|\Delta w\|_0 < n-1$, then the graph associated with Δw is disconnected, and $\Delta f^{(0,\eta)} = \eta \|\Delta w\|_0 \geq 0$. If $\|\Delta w\|_0 \geq n-1$, then $\Delta f^{(0,\eta)} \geq \eta \|\Delta w\|_0 - 1$, which is also non-negative by the assumption $\eta \geq \frac{1}{n-1}$. Then, if $\eta \geq \frac{1}{n-1}$, $\Delta f^{(0,\eta)}$ is non-negative for every $\Delta w \neq \mathbf{0}_m$, hence $w = \mathbf{0}_m$ is an optimal solution to Problem FMMC-$l_0(\eta)$. If $\eta > \frac{1}{n-1}$, then $\Delta f^{(0,\eta)}$ is positive for every arbitrary admissible $\Delta w \neq \mathbf{0}_m$, hence $w = \mathbf{0}_m$ is the only optimal solution to Problem FMMC-$l_0(\eta)$. □

The following example shows that the bound obtained in Proposition 10 is tight, at least if one does not impose further restrictions on the class of graphs to be considered.

Example 2. *Likewise in Example 1, let $n = 2$ and $m = 1$. Then, again, the matrix $P(w)$ has the expression (3.26) and the eigenvalues 1 and $1 - 2w_1$. Hence, on the subset $(0, 1]$ of admissible solutions to Problem FMMC-$l_0(\eta)$, its objective is $|1 - 2w_1| + \eta$, whereas for $w_1 = 0$, the objective is equal to 1. Hence, $w_1 = 0$ is, respectively, the unique optimal solution to Problem FMMC-$l_0(\eta)$ for $\eta > 1$, one of its two optimal solutions for $\eta = 1$ (the other one being $w_1 = \frac{1}{2}$), and a suboptimal solution for $0 < \eta < 1$ (the optimal one being $w_1 = \frac{1}{2}$).*

Remark 9. *Interestingly, Example 2 demonstrates that the functions $\mu(w_0^\circ(\eta))$ and $\|w_0^\circ(\eta)\|_0$ may be not continuous. Indeed, in this example, it follows that $\mu(w_0^\circ(\eta)) = 0$ for $\eta \in (0, 1)$, and $\mu(w_0^\circ(\eta)) = 1$ for $\eta > 1$. Moreover, $\|w_0^\circ(\eta)\|_0 = 1$ for $\eta \in (0, 1)$, and $\|w_0^\circ(\eta)\|_0 = 0$ for $\eta > 1$. A similar remark about the possible absence of continuity holds for Problem FMMC-$l_1(\eta)$, when one considers the same graph of this example.*

d) Non differentiability of the objective at $w = \mathbf{0}_m$.

Likewise Proposition 5, the next result shows that in general the objective of Problem FMMC-$l_0(\eta)$ is not differentiable at $w = \mathbf{0}_m$.

Proposition 11. Let $n \geq 2$. Then,
i) the second-largest eigenvalue modulus term $\mu(w)$ is nondifferentiable at $w = \mathbf{0}_m$ for $n \geq 3$, and differentiable for $n = 2$;

ii) the l_0-pseudo-norm regularization term $\|w\|_0$ is nondifferentiable at $w = \mathbf{0}_m$;
iii) the objective of Problem FMMC-$l_0(\eta)$ is nondifferentiable at $w = \mathbf{0}_m$.

Proof. i) Let us consider at first the case $n \geq 3$. Then, at $w = \mathbf{0}_m$, proceeding likewise in the proof of Proposition 5, the directional derivative of the term $\mu(w)$ in the direction $\underline{e}_i$, i.e.,

$$\lim_{t \to 0^+} \frac{\mu(t\underline{e}_i) - \mu(\mathbf{0}_m)}{t}\,, \tag{3.45}$$

is equal to 0. Now, let $\hat{w}$ be any admissible weight vector with $\hat{w} > \mathbf{0}_m$ (elementwise). Then, at $w = \mathbf{0}_m$, proceeding likewise in the proof of Proposition 5, the directional derivative of the term $\mu(w)$ in the direction $\hat{w}$, i.e.,

$$\lim_{t \to 0^+} \frac{\mu(t\hat{w}) - \mu(\mathbf{0}_m)}{t}\,, \tag{3.46}$$

is different from 0, and differs from

$$\sum_{i=1}^{m} \hat{w}_i \left(\lim_{t \to 0^+} \frac{\mu(t\underline{e}_i) - \mu(\mathbf{0}_m)}{t} \right) = 0 \tag{3.47}$$

(recall Remark 6). Then, we conclude that, for $n \geq 3$, the term $\mu(w)$ is nondifferentiable at $w = \mathbf{0}_m$. Finally, for $n = 2$, the term $\mu(w)$ is differentiable, as shown by Example 2.
ii) Nondifferentiability of the l_0-pseudo-norm term $\|w\|_0$ at $w = \mathbf{0}_m$ follows by the facts that $\|\mathbf{0}_m\|_0 = 0$, and in any neighborhood of $\mathbf{0}_m$ with radius $\varepsilon > 0$ there exists a feasible $w^{(\varepsilon)}$ with $\|w^{(\varepsilon)}\|_0 \geq 1$.
iii) This follows combining the proofs of items i) and ii). □

e) Algorithmic issues.

The following result shows that Problem FMMC-$l_0(\eta)$ is a combinatorial problem and also how solving an instance of such a problem can be reduced to solving several instances of (easier to solve) Problems FMMC. The upper bound $\lfloor \frac{1}{\eta} \rfloor$ of non self-loop edges in the next proposition comes from the consideration that any optimal solution $w_0^\circ(\eta)$ of Problem FMMC-$l_0(\eta)$ cannot have more than $\lfloor \frac{1}{\eta} \rfloor$ non-zero components, otherwise the trivial choice $w = \mathbf{0}_m$ is a better solution.

Proposition 12. Any instance of Problem FMMC-$l_0(\eta)$ can be solved as follows:

– starting from the original graph G associated with the vertex-edge incidence matrix M, generate the set G_η of all its subgraphs G_k with n vertices and with at most $\lfloor \frac{1}{\eta} \rfloor$ non self-loop edges;

– for each subgraph $G_k \in G_\eta$, find an optimal solution $w^\circ(G_k)$ of the instance of Problem FMMC associated with such subgraph, and the corresponding $\mu(w^\circ(G_k))$;

– find a subgraph $G_{k^\circ} \in G_\eta$ that solves the optimization problem

$$\text{minimize}_{G_k \in G_\eta} \left(\mu(w^\circ(G_k)) + \eta \|w^\circ(G_k)\|_0 \right) . \tag{3.48}$$

Then,
i) the resulting vector $w^\circ(G_{k^\circ(\eta)})$ is an optimal solution of Problem FMMC-$l_0(\eta)$;
ii) all optimal solutions of Problem FMMC-$l_0(\eta)$ can be generated according to the procedure above. Moreover, the same optimal solution $w_0^\circ(\eta)$ of Problem FMMC-$l_0(\eta)$ may be generated starting from more than one subgraph $G_{k^\circ(\eta)} \in G_\eta$ (which is optimal for the optimization problem (3.48)), in case of non-uniqueness of the optimal $k^\circ(\eta)$. In particular, one of such subgraphs $G_{k^\circ(\eta)}$ is the one $G_{\tilde{k}(\eta)}$ whose non self-loop edges are the ones associated with the non-zero components of $w_0^\circ(\eta)$.

Proof. i) Since $w^\circ(G_{k^\circ(\eta)})$ is feasible for Problem FMMC-$l_0(\eta)$, one gets, for any optimal solution $w_0^\circ(\eta)$ of Problem FMMC-$l_0(\eta)$,

$$\mu(w^\circ(G_{k^\circ(\eta)})) + \eta \|w^\circ(G_{k^\circ(\eta)})\|_0 \geq \mu(w_0^\circ(\eta)) + \eta \|w_0^\circ(\eta)\|_0 . \tag{3.49}$$

Let us show by contradiction that also the reverse inequality holds in (3.49), implying the equality therein. Suppose to the contrary that

$$\mu(w^\circ(G_{k^\circ(\eta)})) + \eta \|w^\circ(G_{k^\circ(\eta)})\|_0 > \mu(w_0^\circ(\eta)) + \eta \|w_0^\circ(\eta)\|_0 . \tag{3.50}$$

Let $G_{\tilde{k}(\eta)} \in G_\eta$ the subgraph whose non self-loop edges are the ones associated with the non-zero components of $w_0^\circ(\eta)$. Hence, solving Problem FMMC on $G_{\tilde{k}(\eta)}$, one obtains

$$\mu(w^\circ(G_{\tilde{k}(\eta)})) \leq \mu(w_0^\circ(\eta)) , \tag{3.51}$$

by the optimality of $w^\circ(G_{\tilde{k}(\eta)})$ for Problem FMMC on $G_{\tilde{k}(\eta)}$, and the feasibility of $w_0^\circ(\eta)$ for the same optimization problem, and

$$\|w^\circ(G_{\tilde{k}(\eta)})\|_0 \leq \|w_0^\circ(\eta)\|_0 , \tag{3.52}$$

by construction. Hence,

$$\begin{aligned}
&\mu(w^\circ(G_{\tilde{k}(\eta)})) + \eta\|w^\circ(G_{\tilde{k}(\eta)})\|_0 \\
\leq &\mu(w_0^\circ(\eta)) + \eta\|w_0^\circ(\eta)\|_0 \\
< &\mu(w^\circ(G_{k^\circ(\eta)})) + \eta\|w^\circ(G_{k^\circ(\eta)})\|_0 \,, \qquad (3.53)
\end{aligned}$$

which contradicts the optimality of $G_{k^\circ(\eta)}$ for the optimization problem (3.48). Hence, (3.50) cannot hold, and $w^\circ(G_{k^\circ(\eta)})$ solves Problem FMMC-$l_0(\eta)$.

ii) Let us consider any optimal solution $w_0^\circ(\eta)$ of Problem FMMC-$l_0(\eta)$. Likewise in the proof of item ii), considering the subgraph $G_{\tilde{k}(\eta)} \in G_\eta$ whose non self-loop edges are the ones associated with the non-zero components of $w_0^\circ(\eta)$, one obtains both (3.51) and (3.52) with the equality, due to the optimality of $w_0^\circ(\eta)$ for Problem FMMC-$l_0(\eta)$. So, $w_0^\circ(\eta)$ is also generated by the procedure detailed in the statement of the proposition. In particular, it is generated starting from the subgraph $G_{\tilde{k}(\eta)}$. However, in general there may exist also other optimal subgraphs $G_{k^\circ(\eta)}$ for the optimization problem (3.48), which generate the same $w_0^\circ(\eta)$, i.e., such that $w^\circ(G_{k^\circ(\eta)}) = w_0^\circ(\eta)$. □

Due to the combinatorial nature of Problem FMMC-$l_0(\eta)$, unfortunately, the number of instances of subproblems FMMC to be considered in formula (3.48) is in general very large (unless the original graph is "small", or η is large), however we remark that:

- each subproblem FMMC is convex and has a semidefinite programming formulation; each one can be solved through the MATLAB function `fmmc.m` in the CVX package (`http://cvxr.com/cvx/download/`), as already mentioned in Section 3.4.2;

- the number of subproblems FMMC to be considered depends on the parameter η, and is non-increasing with respect to η. In particular, larger values for η (which correspond with a larger desired sparsity) are associated with a smaller number of subproblems;

- some simplifications are possible, making it possible to reduce the number of subgraphs to be considered in formula (3.48). For instance, one can detect and remove all disconnected subgraphs. They

can be detected, e.g., either checking the algebraic multiplicity of the associated Laplacian eigenvalue 1, or applying an algorithm presented in (92), which generates all connected subgraphs of a given graph, with the same number of vertices as in the original graph. Finally, isomorphic subgraphs could be also detected and represented by one single subgraph (although the approach should be in practice limited to subgraphs with a small number of edges, since the graph isomorphism problem belongs to the NP class of computational complexity (93));

– as one possible heuristic to obtain good suboptimal solutions to an instance of Problem FMMC-$l_0(\eta)$, Proposition 12 may suggest to generate a small number of random sparse subgraphs G_k of the original graph with vertex-edge incidence matrix M. Subsequently the associated instances of Problem FMMC have to be solved, and finally, among the obtained optimal solutions $w^\circ(G_k)$, we have to consider the one that minimizes $\mu(w^\circ(G_k)) + \eta\|w^\circ(G_k)\|_0$.

f) Choice of the regularization parameter.

Likewise in Section 3.4.2, the theoretical results presented above justify the following practical rule for choosing the regularization parameter η:

– given a positive integer N and a maximal acceptable increase $\varepsilon > 0$ for the second-largest eigenvalue modulus of P with respect to its optimal value μ°_{FMMC} in Problem FMMC, solve Problem FMMC-$l_0(\eta)$ in correspondence of N values $\eta^{(j)}$ for η such that

  · $0 < \eta^{(1)} < \eta^{(2)} < \ldots < \eta^{(N)} < \frac{1}{n-1}$, and

  · $\mu(w_0^\circ(\eta^{(j)})) \leq \mu^\circ_{FMMC} + \varepsilon$ $(j = 1, \ldots, N)$;

– choose $j^\circ \in \{1, \ldots, N\}$ that maximizes the sparsity $s(w^\circ(\eta^{(j)}))$.

Remark 10. *Differently from the case of Problem FMMC-$l_1(\eta)$ investigated in Section 3.4.2, a final reoptimization step is not needed after finding $w^\circ(\eta^{(j^\circ)})$, since $w^\circ(\eta^{(j^\circ)})$ solves an optimization problem including basically also the sparsity in its objective.*

Finally, a possible way to choose the tolerance parameter ε (which has to be in any case smaller than $1 - \mu^{\circ}_{FMMC}$, again to avoid trivial optimal solutions) is given by formula (3.32), likewise in Section 3.4.2.

g) Interpretation of Problem FMMC-$l_0(\eta)$ as a robust version of Problem FMMC.

Likewise Problem FMMC-$l_1(\eta)$, Problem FMMC-$l_0(\eta)$ has the following interpretation. Let us suppose that, for any given "nominal" choice of the weights w_i ($i = 1, \ldots, m$), one has an "uncertainty" Δw_i such that $|\Delta w_i| \leq \delta U(w_i)$, for some fixed $\delta > 0$, where

$$U(w_i) := \begin{cases} 0 & \text{if } w_i = 0\,, \\ 1 & \text{if } w_i > 0\,. \end{cases} \tag{3.54}$$

Then, likewise in Section 3.4.2 f), one can show that the second-largest eigenvalue modulus $\mu(w + \Delta w)$ is bounded from above as

$$\mu(w + \Delta w) \leq \mu(w) + 2\delta \|w\|_0\,. \tag{3.55}$$

Then, an optimal "robust" choice of the nominal weight vector w is obtained minimizing the objective $\mu(w) + 2\delta\|w\|_0$ on the set of admissible weight vectors w, and is obtained replacing the objective $\mu(w)$ of Problem FMMC with $\mu(w) + 2\delta\|w\|_0$. However, this is equivalent to solving Problem FMMC-$l_0((\eta)$ with the choice $\eta = 2\delta$. Likewise in Section 3.4.2 g), when $\delta \geq 1$, $w = \mathbf{0}_m$ is just an optimal nominal choice for the robust version of Problem FMMC just described, which is consistent with Proposition 10. In this case, however, that proposition shows that $w = \mathbf{0}_m$ is an optimal nominal choice even under a less restrictive condition on δ: namely, for any $\delta \geq \frac{1}{2(n-1)}$.

3.5 Results and Discussion

In this section, we first solve numerically Problems FMMC-$l_0(\eta)$ and FMMC-$l_1(\eta)$ on a toy example, in which both problems can be practically solved in a reasonably small amount of time, then their optimal solutions

can be compared. At the end of the section, we compare the optimal solutions of Problems FMMC and FMMC-$l_1(\eta)$, when both problems are applied to a model of a wireless sensor network with a much larger number of nodes/edges. Finally, some future improvements are discussed.

3.5.1 Comparison of Problems FMMC-$l_1(\eta)$ and FMMC-$l_0(\eta)$

The comparison between the two sparse variations of Problem FMMC is performed on a graph with $n = 8$ vertices and $m = 20$ non self-loop edges, which is shown in Fig. 1. Problem FMMC-l_0 is solved by following the procedure described in Proposition 12. One can observe that, in this case, the number of all subgraphs with n vertices is equal to $2^{20} = 1048576$. However, since we are interested only in connected subgraphs, we first generate all such subgraphs (which have at least $n - 1 = 7$ non self-loop edges, since they must contain at least one spanning tree), then we associate all isomorphic connected subgraphs with a single representative connected subgraph. In this way, a total of 8693 non isomorphic subgraphs is generated, on which Problem FMMC is solved, according to the procedure described in Proposition 12. The comparison between the optimal solutions to Problems FMMC-$l_1(\eta)$ and FMMC-$l_0(\eta)$ is performed by varying the regularization parameter η, and considering different ranges for such a parameter in the two problems, since equal values of the parameter are not directly comparable, as being associated with different regularizations. In particular, for both problems, we consider $N = 100$ different values for the regularization parameter equally spaced inside an interval I_1 for Problem FMMC-$l_1(\eta)$ and an interval I_0 for Problem FMMC-$l_0(\eta)$. From now on, we indicate with $\eta(l_1)$ the regularization parameter associated to Problem FMMC-$l_1(\eta)$, while $\eta(l_0)$ represents the regularization parameter associated to Problem FMMC-$l_0(\eta)$.

As a first step, we solve Problem FMMC on the graph shown in Fig. 1, obtaining its optimal solution w°_{FMMC}, whose second-largest eigenvalue modulus is equal to $\mu(w^{\circ}_{FMMC}) = 0.3786$, and the sparsity is equal to $s(w^{\circ}_{FMMC}) = 0$. Then, we study the optimal solutions achieved by Prob-

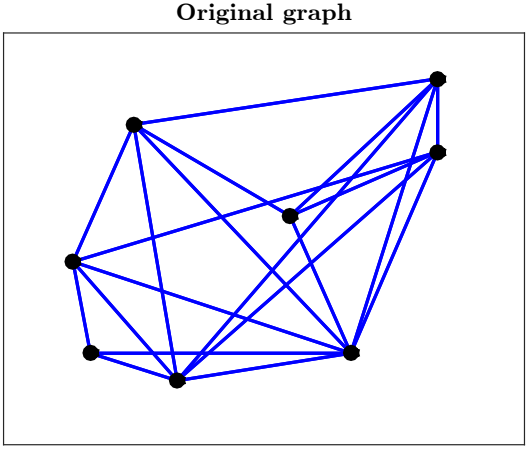

Figure 1: A toy example modeled by a graph with 8 vertices and 20 non self–loop edges.

lem FMMC-$l_1(\eta)$ and Problem FMMC-$l_0(\eta)$. In practice, following the procedures illustrated in Sections 3.4.2 $e)$ and 3.4.3 $f)$, respectively, we aim at determining a feasible solution whose associated second-largest eigenvalue modulus is not much larger than its minimum possible value μ°_{FMMC}, and that, at the same time, provides a satisfactory sparsity. For both procedures, we choose $\rho = 1.5$, which is associated with the tolerance $\varepsilon = 0.145$, as $\mu^{\circ}_{FMMC} = 0.3786$ (see formula (3.32)). Hence, we are interested in studying how the optimal solutions to the two problems vary depending on $\eta(l_1)$ and $\eta(l_0)$, respectively, imposing the upper bound $\mu^{\circ}_{FMMC} + \varepsilon = 0.5236$ on μ. In particular, for Problem FMMC-$l_0(\eta)$ we consider 100 values of $\eta(l_0)$ equally spaced inside the interval $I_0 = [2 \cdot 10^{-8}, 0.08]$; while for Problem FMMC-$l_1(\eta)$ we consider 100 values for $\eta(l_1)$ equally spaced inside the interval $I_1 = [0.02, 0.198]$, since they provide comparable ranges of values for μ at optimality (for graphical reasons, the results in the next figures are reported at a lower resolution).

In Fig. 2, we report, as functions of the regularization parameter, the values of the second-largest eigenvalue modulus μ and the spar-

sity s for the optimal solutions of Problem FMMC-$l_1(\eta)$ (subplots (a) and (b), respectively and Problem FMMC-$l_0(\eta)$ (subplots (c) and (d), respectively). The results in Fig. 2 reveal, as expected, that Problem FMMC–

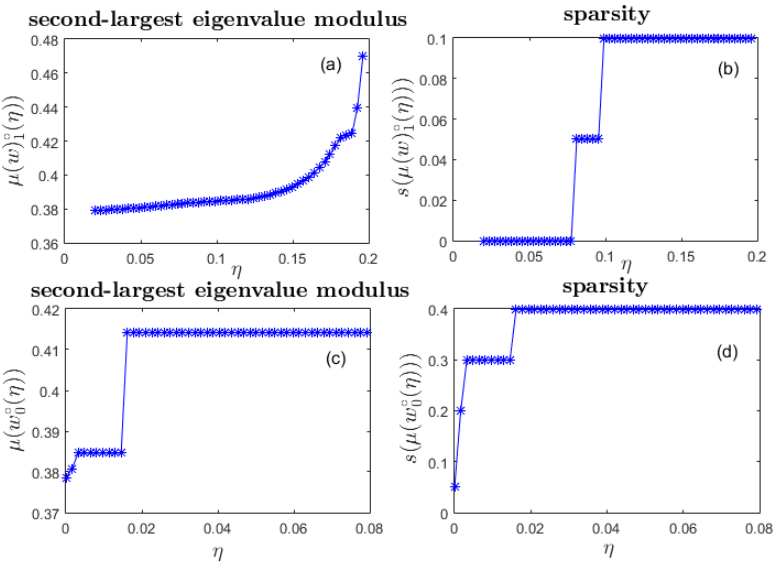

Figure 2: Dependence on the regularization parameter of the second-largest eigenvalue modulus μ and the sparsity s for the optimal solutions of Problems FMMC-$l_1(\eta)$ (subplots (a) and (b)) and FMMC-$l_0(\eta)$ (subplots (c) and (d)).

$l_0(\eta)$ usually provides better solutions than Problem FMMC-$l_1(\eta)$. In fact, in the two respective ranges of values for the regularization parameter, the values of the second-largest eigenvalue modulus obtained solving Problem FMMC-$l_0(\eta)$ are comparable with the ones achieved solving Problem FMMC-$l_1(\eta)$, but with a better sparsity. However, from a computational point of view, solving Problem FMMC-$l_0(\eta)$ for the specific example takes a much longer time than solving Problem FMMC-$l_1(\eta)$ for the same example, i.e., about 40 seconds are needed to solve Problem FMMC-$l_1(\eta)$ for all the 100 values of its regularization parameter, whereas more than 4000 seconds are required to solve Problem FMMC-

$l_0(\eta)$ for all the 100 values of its regularization parameter, since this requires solving also all its subproblems FMMC (one time each). The numerical simulations have been performed using MATLAB R2015a on a notebook with a 1.60 GHz CPU and 8 GB of RAM.

In order to perform another comparison between the two approaches, we also proceed in the following way:

1. we fix a positive integer N_g, then we extract randomly N_g subgraphs over the total of 8683 non isomorphic connected subgraphs of the original graph. This number of subgraphs is chosen in order to be able to find an approximate solution to Problem FMMC-$l_0(\eta)$ in a time comparable to the one needed to solve Problem FMMC-$l_1(\eta)$ exactly (see the next step);

2. we apply a variation of the procedure described in Proposition 12, considering only the subgraphs generated in the step 1) above;

3. we repeat the two steps above for some number N_s of times;

4. we compute the average and standard deviation of the results obtained over the N_s repetitions.

In the following, for illustrative purposes, we always choose $N_g = 100$. We first consider the results achieved by the procedure described above when fixing $N_s = 1$. Fig. 3 shows the values of the second-largest eigenvalue modulus (subplot (a)) and of the sparsity for the suboptimal solution (subplot (b)) to Problem FMMC-$l_0(\eta)$ obtained in this case. Also in this case, the values of the sparsity obtained are better than the ones achieved solving Problem FMMC-$l_1(\eta)$, but larger values of the second-largest eigenvalue modulus are obtained compared with the exact solution of Problem FMMC-$l_0(\eta)$. In addition, when $\eta(l_0)$ is larger than 0.04, the obtained suboptimal solutions do not even satisfy the required constraint $\mu \leq 0.5236$.

We now consider the case $N_s = 10$. The plot on the top of Fig. 4 shows the average and standard deviation of the second-largest eigenvalue modulus of the suboptimal solution to Problem FMMC-$l_0(\eta)$ (subplot (a), whereas subplot (b) does the same for the sparsity. Again, when

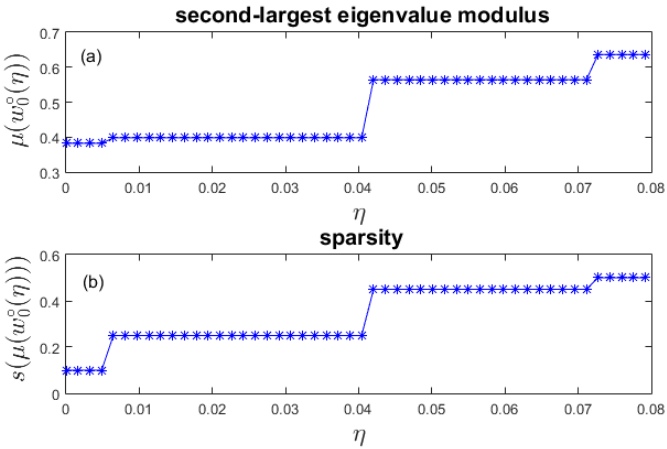

Figure 3: Dependence on the regularization parameter of the second-largest eigenvalue modulus μ and the sparsity s for the suboptimal solution of Problem FMMC-$l_0(\eta)$, obtained when 100 subgraphs are randomly extracted from the whole set of connected non isomorphic subgraphs.

$\eta(l_0)$ is larger than 0.06, in general the obtained suboptimal solutions do not even satisfy the required constraint $\mu \leq 0.5236$. In addition, due to the 10 repetitions, the time needed to obtain these results is about 10 times larger than the one needed to solve Problem FMMC-$l_1(\eta)$ exactly.

3.5.2 Comparison of Problems FMMC-$l_1(\eta)$, FMMC$_{\text{constr}}$-$l_1(\eta)$, and FMMC

We now investigate numerically the optimal solutions of Problems FMMC-$l_1(\eta)$ and FMMC$_{\text{constr}}$-$l_1(\eta)$, comparing them with the one of Problem FMMC. In particular, as a test example, we consider a vertex-edge incidence matrix M corresponding to a model of a wireless sensor network with 50 vertices and 200 edges, generated in a similar way as the one in (82, Section 5.1). The first two plots in Fig. 5 (subplots (a) and (b), respectively), which refers to the behavior of an optimal solution $w_1^\circ(\eta)$ with respect to η, confirm the statement of Proposition 3 about the opposite monotonic dependence on η of $\mu(w_1^\circ(\eta))$ and $\|w_1^\circ(\eta))\|_1$. Subplot

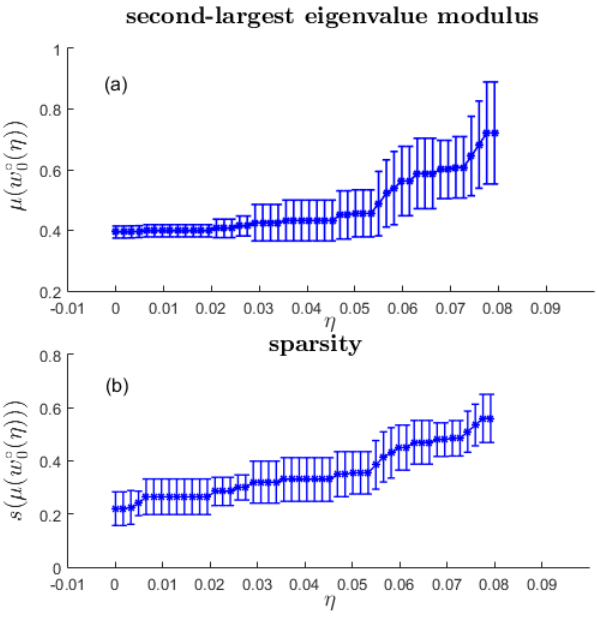

Figure 4: Dependence on the regularization parameter of the average (over 10 trials) of the second-largest eigenvalue modulus μ (upper plot) and of the sparsity s (lower plot) for the suboptimal solution of Problem FMMC-$l_0(\eta)$, obtained when 100 subgraphs are randomly extracted from the whole set of connected non isomorphic subgraphs.

(c) shows its sparsity $s(w_1^\circ(\eta))$ as a function of η, which in this particular case is not a monotonic function of η. However, the plots also show that $w_1^\circ(\eta)$ is sparser than the optimal solution of Problem FMMC, for all the considered values of η. So, they highlight the possibility, in this case, of finding a value of the parameter η for which the second-largest eigenvalue modulus $\mu(w_1^\circ(\eta))$ is not much larger than its minimum possible value μ_{FMMC}°, and that, at the same time, provides a satisfactory sparsity of $w_1^\circ(\eta)$. Again, in order to find such a parameter, we follow the procedure illustrated in Section 3.4.2 e). We choose $\rho = 1.5$, associated with the tolerance $\varepsilon = 0.027$, as $\mu_{FMMC}^\circ = 0.9165$ in this particular case. We also consider $N = 20$ values $\eta^{(1)}, \ldots, \eta^{(N)}$ for the regularization parameter η

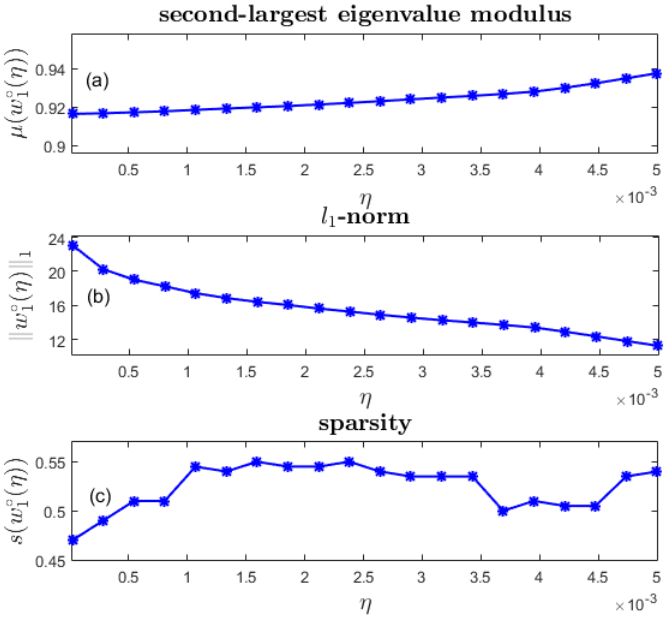

Figure 5: Dependence on η of the second–largest eigenvalue modulus of the weighted adjacency matrix P, the l_1–norm and the sparsity of an optimal solution of Problem FMMC–l_1.

(uniformly spaced in the interval $[2 \cdot 10^{-5}, 5 \cdot 10^{-3}]$, see Fig. 5), obtaining $j^\circ = 5$ and $\eta^{(j^\circ)} = 1.1 \cdot 10^{-3}$ as the optimal regularization parameter. For this value, we obtain $\mu(w_1^\circ(\eta^{(j^\circ)})) = 0.9186$, $\|w_1^\circ(\eta^{(j^\circ)})\|_1 = 17.45$, and $s(w_1^\circ(\eta^{(j^\circ)})) = 0.545$. Compared with the optimal solution w_{FMMC}° of Problem FMMC (for which $\mu(w_{FMMC}^\circ) = 0.9165$, $\|w_{FMMC}^\circ\|_1 = 23.71$, and $s(w_{FMMC}^\circ) = 0.41$), the increase of the second-largest eigenvalue modulus, the decrease of the l_1-norm of the weight vector, and the increase of its sparsity are, respectively, about $0.2\%, 26\%$, and 25%. In terms of the mixing time (3.4), we obtain an increase of about 3% with respect to the value associated with w_{FMMC}°. Fig. 6 shows: the original graph associated with the given vertex-edge incidence matrix M (subplot (a));

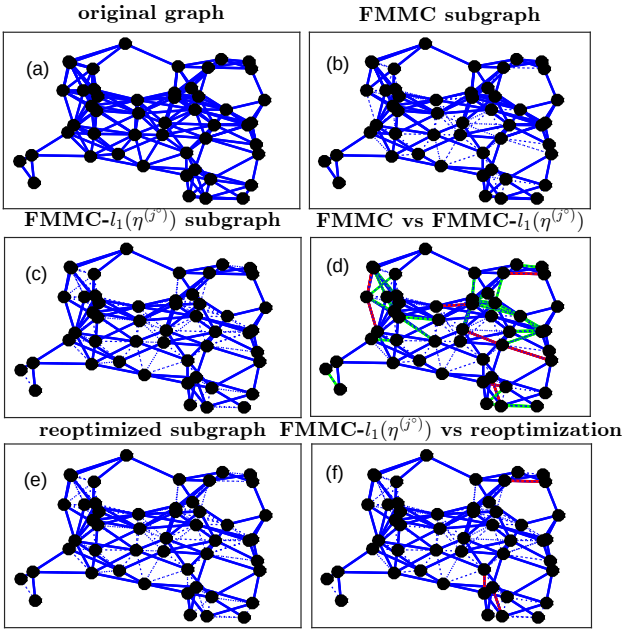

Figure 6: A comparison of the subgraphs associated with non-zero weights in the optimal solutions to Problems FMMC and FMMC-$l_1(\eta^{(j^\circ)})$. See the main text for explanations about the colors used in the figure.

its subgraph obtained keeping only the edges associated with non-zero weights in the optimal solution w°_{FMMC} to Problem FMMC (subplot (b)); the one obtained keeping only the edges associated with the non-zero weights of $w^\circ(\eta^{(j^\circ)})$ (subplot (c)); a comparison of the two subgraphs (subplot (d)), obtained merging such subgraphs and highlighting in blue the non-zero-weighted edges appearing in both graphs and in green (resp., red) the non-zero weighted edges of the optimal solution to Problem FMMC (resp., Problem FMMC-$l_1(\eta^{(j^\circ)})$) that are associated with zero weights in the optimal solution to Problem FMMC-$l_1(\eta^{(j^\circ)})$ (resp., Problem FMMC). In particular, starting from the original 200 edges joining different vertices, the optimal solution to Problem FMMC keeps 118 edges, while the optimal solution to Problem FMMC-$l_1(\eta^{(j^\circ)})$ keeps only 91 edges. The percentage of edge reduction when moving from w°_{FMMC}

to $w_1^\circ(\eta^{(j^\circ)})$ is therefore about 23%.

As described in Section 3.4.2 e), after finding the parameter $\eta^{(j^\circ)}$, an additional improvement may be obtained performing a reoptimization step, solving Problem FMMC on the sparser subgraph obtained deleting the edges associated with zero weights in the obtained optimal solution $w_1^\circ(\eta^{(j^\circ)})$ to Problem FMMC-$l_1(\eta^{(j^\circ)})$. This step is illustrated in subplots (e) and (f) of Fig. 6, which shows in red the edges deleted by the reoptimization step. In this way, a new weight vector w_{reopt}° is obtained with $\mu(w_{\text{reopt}}^\circ) \leq \mu(w_1^\circ(\eta^{(j^\circ)}))$ and $s(w_{\text{reopt}}^\circ) \geq s(w_1^\circ(\eta^{(j^\circ)}))$. So, compared with $w_1^\circ(\eta^{(j^\circ)})$, the sparsity of the weight vector w_{reopt}° either remains the same or even increases, whereas the second-largest eigenvalue modulus either remains the same or even decreases. Indeed, after the reoptimization step, we obtain $\mu(w_{\text{reopt}}^\circ) = 0.9169$ and $s(w_{\text{reopt}}^\circ) = 0.56$.

Finally, we report the results obtained solving Problem FMMC$_{\text{constr}}$-$l_1(\eta)$ (in this case, for simplicity of comparison, for $\eta = \eta^{(j^\circ)}$), imposing the constraint that the 11 non self-loop edges associated with the vertex A in Fig. 7 are fixed, resp., to the values $0.1, 0.05, 0.25, 0.1, 0.01, 0.07, 0.05, 0.1, 0.02, 0.05, 0.1$, whose sum is $0.9 < 1$ (hence the problem is feasible). Since such constraints are not satisfied by $w_1^{(j^\circ)}$, a significant change of the optimal solution is expected, with respect to the unconstrained version of the problem. Indeed, for the optimal solution $w_{\text{free},1}^\circ(\eta^{(j^\circ)})$ to such an instance of Problem FMMC$_{\text{constr}}$-$l_1(\eta^{(j^\circ)})$, we obtain $\mu(w_{\text{free},1}^\circ(\eta^{(j^\circ)})) = 0.9193$. Such an increase of $\mu(w_{\text{free},1}^\circ(\eta^{(j^\circ)}))$ with respect to $\mu(w_1^\circ(\eta^{(j^\circ)}))$ was also expected, as Problem FMMC$_{\text{constr}}$-$l_1(\eta^{(j^\circ)})$ is more constrained than Problem FMMC-$l_1(\eta^{(j^\circ)})$, and has the same objective (including in the objective also the fixed weights).

3.5.3 Discussion and Conclusions

In this work, we have presented some theoretical and numerical results about several sparse variations of the Fastest Mixing Markov–Chain Problem. Among possible future developments we mention the possibility of using other sparsity enforcing regularization terms (such as the reweighted l_1–norm (94), the group LASSO (95) and the sparse group

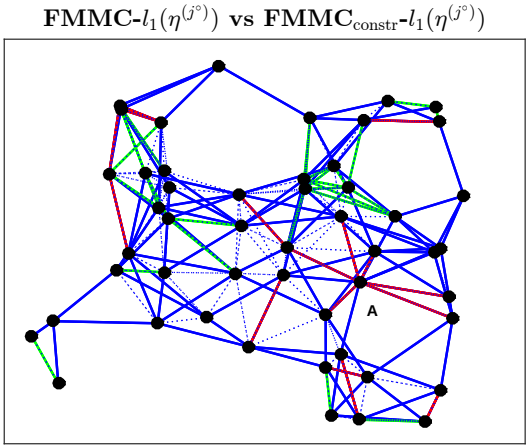

Figure 7: A comparison of the subgraphs associated with non-zero weights in the optimal solutions to Problems FMMC-$l_1(\eta^{(j^\circ)})$ and FMMC$_{\text{constr}}$-$l_1(\eta^{(j^\circ)})$. This is obtained by merging such subgraphs and highlighting in blue the non-zero-weighted edges appearing in both graphs and in green (resp., red) the non-zero weighted edges of the optimal solution to Problem FMMC$_{\text{constr}}$-$l_1(\eta^{(j^\circ)})$ that are associated with zero weights in the optimal solution to Problem FMMC-$l_1(\eta^{(j^\circ)})$ (resp., Problem FMMC$_{\text{constr}}$-$l_1(\eta^{(j^\circ)})$).

LASSO (96)). At the same time it would be useful to solve the proposed regularized optimization problems in a distributed way. Finally we mention the possibility of extending the theoretical analysis to nonlinear and stochastic consensus problems.

3.6 Summary

In this chapter we described sparse variations of the consensus problem with the aim of determining solutions that lead to a fast convergence to the consensus state keeping, at the same time, the network of the agents as sparse as possible. In particular we started from Problem FMMC which is concerned with the study of the topology of the network of agents with the aim of reaching the consensus state as fast as possible. We presented theoretical statements and satisfactory numerical

results for both the sparse variations introduced (the one involving the l_1 norm of the variables vector and the one with the l_0–pseudo norm of the variables vector).

In the next chapter we will continue the study of the consensus problem, paying particular attention on the structure of the graph of interconnections G. We will deal with the problem of increasing the rate of convergence to the consensus state from another point of view. In fact, in the approach presented in the next chapter, we are not interested in determining the optimal entries of P that minimize $\mu(P)$, but we intend to "decompose" opportunely the original network G in different subgraphs where the rate of convergence to the consensus state should be faster. We intend to define a sort of hierarchical procedure that can be easily parallelized and can increase the overall rate of convergence to the common opinion.

4

Spectral graph theory in the consensus problem

4.1 Introduction

We continue our discussion on the consensus problem with the aim of better evaluating and studying the spectral properties of the agents network. Our idea is to develop an approach able to increase the convergence rate to the consensus state by dividing the original graph in many subgraphs where the rate of convergence to the "local" consensus state (i.e., the consensus state when the consensus algorithm is run on the subgraphs) should be faster. Our approach consists in extracting subgraphs with "good" spectral properties that lead to a fast convergence to the local consensus state. We intend to apply both a spectral clustering algorithm and another technique, developed for this study, that should be able to extract "dense" subgraphs for which the second–largest eigenvalue modulus of the transition probability matrix is relatively small and

This chapter is partly based on:

- Rita Morisi, Giorgio Gnecco, Alberto Bemporad "A hierarchical consensus method for the approximation of the consensus state, based on clustering and spectral graph theory", submitted to Engineering Applications of Artificial Intelligence.

the corresponding rate of convergence to the consensus state relatively fast. Similar approaches to the one we intend to study have been proposed in (97; 98). They propose to decompose the multi–agent system in an optimal hierarchical structure considering a multi–layer method. Nevertheless, in (98) they approximate the rate of convergence to the consensus state by evaluating its upper bound based on the Laplacian matrix of the network, while, as we will see later, we ground our study on the transition probability matrix. In addition, neither (97) nor (98) consider techniques that exploit the spectral properties of the graph for the detection of different subgraphs. We thus believe that, compared with (97; 98) the main original contribution of the present chapter lies on the techniques adopted to determine the different connected subgraphs and on the theoretical study using the Cheeger's inequality (Formula (2.7)) to explain and prove the results shown in Section 4.5. In Section 4.2 we first provide the formulation of the specific problem, then in Section 4.3 we focus on the methods adopted to divide the original graph. Section 4.4 provides a theoretical study of the time of convergence to the consensus state of the method developed, while Section 4.5 presents numerical examples. In Section 4.6 we better evaluate from both a theoretical and practical point of view a problematic issue that arises in some situations and we subsequently provide a solution able to improve the final results. Section 4.7 offers the conclusions and the final discussion.

4.2 Problem formulation

In Chapter 3 the consensus problem has been studied starting from the approach developed in (48), where, for a given graph topology, it has been solved through a convex optimization problem able to determine the entries of the transition probability matrix P that minimize its second–largest eigenvalue modulus. Now, we are not interested in optimizing the entries of the matrix P, but we intend to increase the convergence rate to the consensus state by considering local consensus subproblems run on different subgraphs of the original network and then to determine the final consensus state by evaluating every single local con-

sensus state. Our method consists of two steps: the first step can be easily parallelized, hence, it should be able to increase the rate of convergence to the consensus state. This phase consists in dividing the original graph $G = (V, E)$ with $|V| = N$ agents in many subgraphs (evaluating the spectral properties of the network) where the consensus algorithm can be run in parallel. The second step, instead, consists in determining an auxiliary graph that connects, depending on the connection in the original graph, some selected nodes of the previous subgraphs. The consensus state determined on this auxiliary graph is the same, up to a certain tolerance, of the one determined in the original network (details are provided in the following sections). We ground our analysis of the convergence rate to the consensus state on the following upper bound (see e.g., (99)):

$$\|x(t) - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T x(0)\|_2^2 \leq \mu(P)^t \|x(0)\|_2^2. \tag{4.1}$$

Formula (4.1) is used to determine an approximation of the rate of convergence to the consensus state on each graph/subgraph considered. In fact, defining a suitable probability matrix of interconnections P on a connected graph (condition required by the consensus problem in order to have all the agents agree upon a common opinion), Formula (4.1) can be applied both to the original graph G and to different subgraphs obtained from G. The entries p_{ij} and the dimension of P change depending on the graph/subgraph considered. We aim at exploiting Formula (4.1) in order to find subgraphs with "good" spectral properties, i.e., with fast convergence rate to the consensus state inside each single subgraph. We recall that $\mu(P)$ in Formula (4.1) is the second–largest eigenvalue modulus of P associated to the dynamical system (now every matrix P associated to each graph/subgraph is determined a–priori), $x(t)$ is the vector of the different opinions of the agents at time t, $x(0)$ is the vector of the initial opinions and $\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T x(0)$ is the consensus state, i.e., the average of the initial opinions of the agents. $x(0)$, thus $\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T x(0)$, change depending on the graph/subgraph considered. When referred to a subgraph extracted from G, $x(0)$ contains the initial opinions only of the agents associated to the subgraph considered. Again, we are dealing with undirected graphs, hence with symmetric transition probability matrices that

lead to the average consensus problem (i.e., the consensus state is the average of the initial opinions of the agents).

As stated in the previous chapter, from the previous inequality we can infer that the smaller the second–largest eigenvalue modulus $\mu(P)$, the faster the convergence rate to the consensus state. Our idea is to determine a method able to decrease the value of the second–largest eigenvalue modulus $\mu(P)$ by choosing suitable subgraphs, and exploiting the inequality (4.1) that gives an estimate of the time t needed by the system to reach the consensus state up to a fixed tolerance. We proceed our study by first describing the method used to compute the transition probability matrix of a graph starting from its topology. Subsequently, we describe in details how different subgraphs are extracted from the original network.

4.2.1 Computation of the transition probability matrix

The method briefly reported below, that is used to compute the transition probability matrix P involved in the linear system (3.1), is described in (100). In particular, given the unweighted adjacency matrix $A \in \mathbb{R}^{N\times N}$ of a graph, a doubly stochastic and symmetric matrix P can be computed determining a diagonal matrix

$$\Delta = \begin{bmatrix} \delta_1 & 0 & \cdot & \cdot & 0 \\ 0 & \delta_2 & \cdot & \cdot & 0 \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ 0 & & & & \delta_N \end{bmatrix},$$

hence the unknown diagonal entries δ_i with $i = 1, \ldots, N$, such that

$$P = \Delta + \epsilon(A - I);$$

where I is the identity matrix. Thus, choosing ϵ such that $0 < \epsilon < \frac{1}{d_{\max}}$, where $d_{\max}$ is the maximum node degree of the graph, the solution of a system of N equations with N unknowns equal to the diagonal elements of Δ gives the desired matrix P with entries

$$p_{ij} = \begin{cases} \epsilon & \text{if } (i,j) \in E \quad \text{and} \quad i \neq j \\ 1 - \epsilon d_i & \text{if } i = j \end{cases}$$

where d_i is the degree of node i. In particular, the condition on ϵ is necessary to guarantee that all the diagonal terms are positive. The procedure just described is used to compute the transition probability matrix P of the original graph G and of all the subgraphs (corresponding to the subsets of agents extracted) involved in the problem. When computing the matrix P associated either to the original graph G or to a generic subgraph extracted from G, we fix $\epsilon = \frac{1}{d_{\max}+1}$, where $d_{\max}$ is the maximum degree of the specific graph/subgraph. In this way, the constraint $0 < \epsilon < \frac{1}{d_{\max}}$ always holds.

4.3 Dividing graph G to increase the convergence rate to the consensus state

We aim at dividing the original graph $G = (V, E)$ in different connected subgraphs with a smaller second–largest eigenvalue modulus than the one associated to the original network. On each subgraph we estimate the time needed by the agents involved in that subgraph to reach the local consensus state up to a certain tolerance evaluating the upper bound in formula (4.1). Later we will see that subgraphs with nodes in common are allowed. However, this does not create a problem to our procedure that can be easily parallelized. After each subgraph has reached an approximation of its own consensus state, an auxiliary network with a number of nodes equal to the number of subgraphs previously generated is created. To each node of the auxiliary graph is associated the value of the local consensus state previously computed. The consensus algorithm is run again on the auxiliary graph and the upper bound in (4.1) is used to evaluate the convergence rate to the consensus state on this final auxiliary graph. In particular, the consensus state determined with the procedure just described is the same, up to a certain tolerance, of the one obtained directly applying the algorithm to the original graph (see e.g., section 4.4). As previously mentioned, two consecutive steps are run: the first one consists in identifying a set of subgraphs with "good" spectral properties, the second one aims at building a final auxiliary graph where the consensus algorithm is run again in or-

der to determine the final consensus state; this is a sort of hierarchical method. Figure 8 explains with a simple example the approach just described. In this case, the convergence rate of the consensus algorithm directly applied to the original graph, the one in the upper left corner, is slow. In fact, the information related to the different opinions slowly flows, for instance, from the group of agents in red and either the group of agents in yellow or blue since these groups are poorly connected (indeed, there is only one edge among the blue/yellow agents and the red ones). However, extracting a set of denser subgraphs may lead to a faster convergence rate to the local consensus state in each subgraph, since in that case the information would flow faster than in the original graph (for the example considered, the ideal case would be associated with the extraction of the 3 subgraphs shown with different colors in Figure 8). Then, in the specific case, the second phase deals with a small and sufficiently dense auxiliary graph, for which the rate of convergence to the associated consensus state is fast. In this example we suppose to extract the 3 subgraphs with different colors such that in the $2nd$ phase each one is represented by only one node that summarizes the opinions of all its agents. Each node of the auxiliary graph used in the $2nd$ phase has an initial value equal (up to a given tolerance) to the average of the initial opinions of the agents of the subgraph it comes from. Concerning the first step, we consider two techniques able to divide the graph in many subgraphs; the first one consists in applying a spectral clustering algorithm to the original dataset; we provide a brief summary of this technique in the following section. The second one, instead, has been developed for this particular study, and it is described in section 4.3.2. We call this procedure *nearest supernode approach*.

We indicate with M the number of subgraphs we intend to define with both the spectral clustering and the *nearest supernode approach* used during the $1st$ step. Thus, from the original network G we determine M different subgraphs G_m with $m = 1, \dots, M$. The goal of the hierarchical method is to generate the subgraphs G_m such that the rate of convergence to the consensus state inside each subgraph is faster than the one in the original graph G. The choice of the two clustering techniques

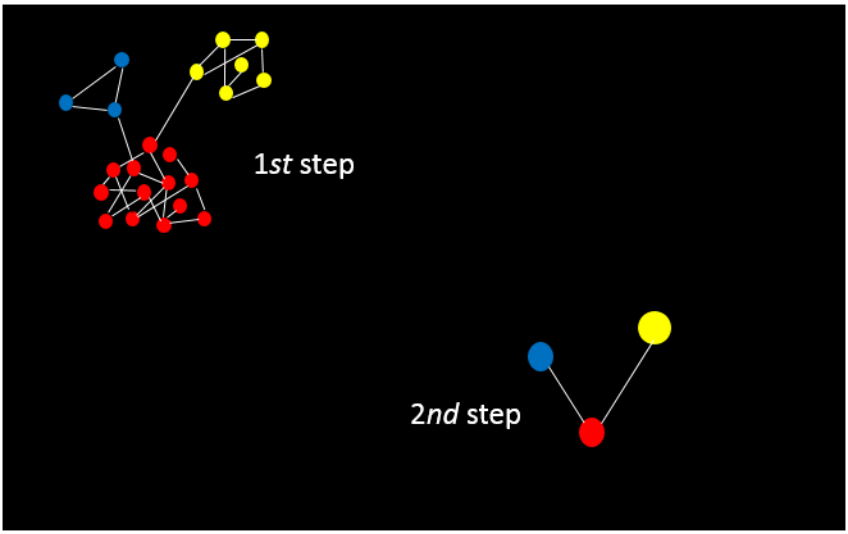

Figure 8: For a simple example: the subgraphs and the auxiliary graph determined, respectively, in the first phase and in the second phase of the proposed hierarchical consensus method.

described in the following subsections reflects this goal, since they are expected to generate "dense" subgraphs. It is worth remarking that, by construction, also the auxiliary graph G_{aux} is expected to be dense. Indeed, due to Formula (2.8), its volume is equal to the number of subgraphs (hence, it is small when this number is small), whereas its diameter is expected to be small, due to the rule used for the construction of its edges.

4.3.1 Spectral clustering

As described in Section 2.3.1, spectral clustering is a technique able to determine the subgraphs inside an original graph, by exploiting the eigenvalues and eigenvectors of the Laplacian matrix of the graph. In particular, this kind of method needs the number of clusters, the M subgraphs in our problem, one intends to detect inside the original graph; thus, if one has knowledge about the structure and the topology of the

original graph, it is easier to provide to the algorithm the information about the number M of clusters/subgraphs it is expected to exist. For our study we intend to apply this method to the network of agents by evaluating different numbers of clusters/subgraphs we require the method to detect. In fact, we test the hierarchical method on both graphs with and without a clear clustering structure; thus, we try different options in order to understand the ones that lead to the best results. In particular we use an algorithm that makes use of the normalized Laplacian (see e.g., (60)).

4.3.2 *Nearest supernode approach*

Beside the spectral clustering algorithm, we intend to develop another method for the determination of the subgraphs G_m, with $m = 1, \ldots, M$. In fact, we would like to define a more automatic technique (no knowledge about the structure of the graph is required) and less expensive from a computational point of view than the spectral clustering algorithm. Indeed, especially for the case of networks with a huge number of agents, the application of spectral clustering can be difficult, since this method requires the computation of selected eigenvectors of the normalized Laplacian matrix, whose number of elements grows quadratically with the number of agents. Hence, here we propose a second clustering method, which we call *nearest supernode approach*, able to overcome this issue, and to produce results comparable with the ones achieved by spectral clustering (a numerical comparison of the two methods, confirming this expectation, is reported later in Section 4.5).

The following rules are used by the proposed *nearest supernode approach* to determine suitable subgraphs G_m of the original connected graph G.

- Starting from a connected graph G, the number M of subgraphs considered to partition G is fixed;
- M nodes, called *supernodes*, are generated; they are used to create the *subgraphs*. In the next sections, these *supernodes* are generated

following 4 different types of procedure: either they can be randomly sampled from the set V (without repetition) or they can be selected depending on different measures. More precisely, we can choose to consider the first M nodes with the highest degree, with the highest betweenness centrality, or with the highest clustering coefficient (see e.g., Section 2.2.1 for a description of these graph features);

- the set of nodes V is partitioned into two disjoint subsets SN and ON, where SN is the set of *supernodes* determined above, while $ON = V \setminus SN$ is the set containing the other nodes of the graph G;

- the nodes in ON are assigned to the *supernodes* in the following way:

  - a node v_i from ON is randomly picked up;
  - the shortest path (SP) (based on the unweighted adjacency matrix of the graph) between v_i and all the supernodes is computed;
  - three possible situations can arise: v_i has $\text{length}(SP) = 1$ to only one *supernode*, thus, it is assigned to this *supernode*; different *supernodes* with $\text{length}(SP) = 1$ to v_i exist, v_i is then randomly associated to one of the nearest *supernodes*. Finally, no *supernodes* with $\text{length}(SP) = 1$ to v_i exist. Thus, v_i is either assigned to the nearest *supernode* with $\text{length}(SP) > 1$ or randomly assigned to one of the nearest *supernodes* with $\text{length}(SP) > 1$, if more than one nearest *supernodes* exist. In this situation, all the nodes in the path are associated to the nearest *supernode*; this prevents subgraphs to be disconnected and it can allow some nodes to be shared by different subgraphs (details are provided below). Note that, each node belonging to subset ON is associated to a node in SN depending on the shortest path without imposing constraints on the number of nodes associated to the *supernodes*. Thus, subgraphs with different dimensions are generated; in gen-

eral, *supernodes* with high degree are expected to generate subgraphs with larger dimension than *supernodes* with low degree.

In order to allow each subgraph to converge to its own consensus state, subgraphs G_m have to be connected; thus, the procedure just described allows the presence of overlaps, i.e., nodes shared by different subgraphs. As just described, the subgraphs are determined by randomly choosing a node $v_i \in ON$ and then assigning it to the nearest *supernode*. If the length of the shortest path SP between v_i and every single *supernode* is greater then 1, then v_i is assigned to the nearest *supernodes* with $\text{length}(SP) > 1$. To avoid the presence of isolated nodes, all the nodes in the shortest path are associated to the same *supernode*. Thus, let us consider the situation depicted in Figure 9. The nodes with indexes 2 and 7 have been chosen as *supernodes*. Thus, $SN = \{2, 7\}$, while $ON = \{1, 3, 4, 5, 6, 8\}$. Nodes with indexes 1 and 3 are associated to 2 since they have $\text{length}(SP) = 1$ to the *supernode* 2, while nodes 6 and 8 are associated to 7. Node indexed 4 is then picked up and randomly assigned to one of the two *supernodes* since it has $\text{length}(SP) = 1$ to both of them. Let us suppose that it has been assigned to node 7; up to now subgraph $G_1 = \{1, 2, 3\}$, while $G_2 = \{4, 6, 7, 8\}$. Finally, node with index 5 is picked up and again it is randomly assigned to one of the two supernodes ($\text{length}(SP) = 2$ to both of them). If *supernode* 2 is chosen between the two, then node with index 4 is associated to subgraph G_1 too. Thus, node with index 4 is shared between subgraph G_1 made of the nodes in yellow and node with index 2 and subgraph G_2 with the blue nodes and node indexed 7; with this procedure no isolated nodes are allowed. The presence of nodes in common between different subgraphs does not represent a problem since the shared nodes can be seen as multiple independent copies of the same node in different subgraphs.

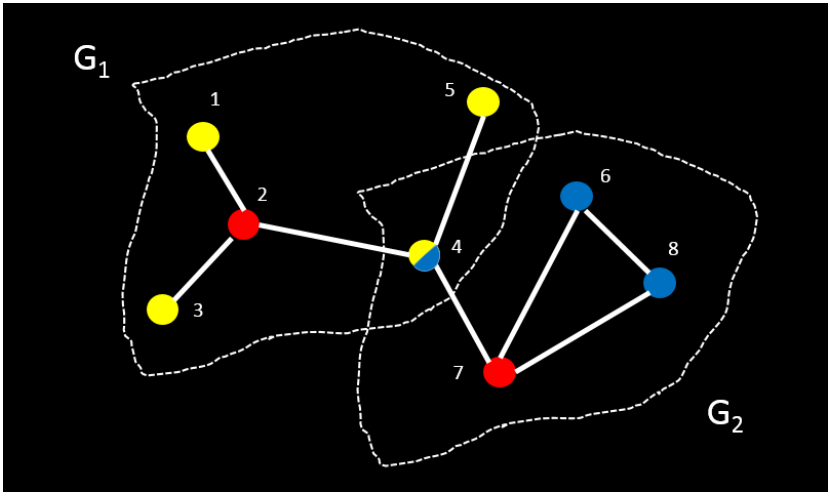

Figure 9: Examples of subgraphs generation where one node is shared between two subgraphs.

4.4 Consensus on the subgraphs and on the auxiliary graph

For each type of graph G studied (they will be better described later) both the methods described above are used to extract the M subgraphs G_m with $m = 1, \ldots, M$. Once the subgraphs have been generated, each subgraph evolves independently according to Formula (3.1) (with obvious changes in notation, to adapt it to that subgraph). This phase requires a number of iterations of Formula (3.1) sufficiently large to allow all the subgraphs to approximate the local consensus state within a desired accuracy. This number of steps is approximately equal to the one required by the subgraph $G_{\hat{m}}$ with the largest $\mu(P_{\hat{m}})$ among the second–largest eigenvalue moduli $\mu(P_m)$ of all the subgraphs, because such subgraph is the one with the smallest rate of convergence to the local consensus state (see Formula (3.3)). After the subgraph $G_{\hat{m}}$ has reached the desired approximation of its local consensus state, the second phase of the hierarchical consensus method starts, by determining an auxiliary graph

G_{aux} with a number of nodes equal to the number of subgraphs previously determined. The nodes in this graph are connected depending on the connections in the original graph. More precisely, two nodes v_i and v_j of G_{aux} are connected by an edge in G_{aux} if and only if at least two nodes of G belonging to the subgraphs associated, respectively, with v_i and v_j, are connected by an edge of G. Once the auxiliary graph is built, it also evolves according to Formula (3.1) (with obvious changes in notation, to adapt it to the auxiliary graph), and an approximation of its consensus state is determined, after some number of iterations. With a proper initialization of the states of the nodes belonging to the auxiliary graph (see Section 4.4.3), this is also an approximation, up to a desired tolerance, of the global consensus state of the original graph G. It is worth remarking that, when the *nearest supernode approach* is used in the first phase of the hierarchical consensus method, all the subgraphs G_m and the auxiliary graph G_{aux} are guaranteed to be connected. Spectral clustering, instead, does not provide such guarantee. Nevertheless, when spectral clustering is used, the subgraphs G_m and the auxiliary graph G_{aux} are usually connected, at least for small choices of M. In the following, we assume that all the graphs/subgraphs are connected.

In the next subsection, we provide a detailed analysis of conditions under which, the desired approximation of the global consensus state of G being the same, the proposed hierarchical method requires a total number of iterations smaller than the one required by the direct evaluation of Formula (3.1) on the original graph G.

4.4.1 Approximation of the global consensus state through the hierarchical method

We recall that we associate with the graph G, with each subgraph G_m, and with the auxiliary graph G_{aux}, respectively, the transition probability matrices P, P_m, and P_{aux}, which are all generated according to the procedure described in Section 4.2.1. Such a procedure, indeed, is guaranteed to generate a doubly stochastic and symmetric matrix for which formula (3.2) holds. By the short-hand notations μ, μ_m, and μ_{aux}, we in-

dicate, respectively, the second–largest eigenvalue modulus of the transition probability matrices P, P_m, and $P_{\rm aux}$.

Since the graphs and subgraphs involved in the hierarchical consensus method have in general different numbers of nodes, it is useful to consider in the analysis the l_∞–norms rather than the l_2–norms of the vectors of initial opinions. In this way, the elements of such vectors are more easily comparable. Moreover, as it is shown later in Section 4.4.3, using the l_∞–norm also allows one to translate some upper bounds valid for the graph G to upper bounds valid for G_m and $G_{\rm aux}$. In order to perform the approximation error analysis using the l_∞–norm, we recall that, given a generic vector $z \in \mathbb{R}^N$, its l_2–norm and l_∞–norm are related by the following inequalities:

$$\|z\|_\infty \leq \|z\|_2 \leq \sqrt{N}\|z\|_\infty \,. \tag{4.2}$$

This, combined with the bound (4.1), provides

$$\left\| x(t) - \frac{1}{N}\mathbf{1}\,\mathbf{1}^T x(0) \right\|_\infty^2 \leq \mu^t(P)|V|\,\|x(0)\|_\infty^2 \,, \tag{4.3}$$

which is the main tool used for the next approximation error analysis (with obvious changes in notations, similar bounds hold for each G_m, and for $G_{\rm aux}$).

The time needed to reach a desired accuracy in the approximation of the consensus state of the graph G through both the non–hierarchical consensus method and the proposed hierarchical consensus method is estimated in the following way:

- a tolerance $\varepsilon > 0$ is fixed; this tolerance represents, for both methods, the desired accuracy in the approximation, in the l_∞–norm, of the consensus state of the original graph G;
- the number of iterations needed by formula (3.1) applied to the original graph G to reach the consensus state up to the tolerance ε is bounded from above by choosing the smallest value T of t for which

$$\mu^t(P)|V|\|x(0)\|_\infty^2 \leq \varepsilon^2. \tag{4.4}$$

(see Formula (4.3)). Note that, to compute T, here we are assuming that $\|x(0)\|_\infty$ is known, but we are not assuming the values of every single element of the vector $x(0)$ to be also known. This assumption could be relaxed replacing $\|x(0)\|_\infty$ in formula (4.4) with an upper bound. However, as shown later in Section 4.4.3, this relaxation is not essential for the analysis;

- a similar kind of performance analysis is applied to every single subgraph, evaluating an upper bound $t_{1^\circ\text{phase}}$ on the number of iterations needed to reach the local consensus state, in this case up to the tolerance $\frac{\varepsilon}{2}$, still with respect to the l_∞–norm. In fact, since the hierarchical method is made of two consecutive phases (the first one with each subgraph evolving independently according to formula (3.1), and the second one involving the auxiliary graph), and since the l_∞–norm is used in the analysis, one can fix a tolerance equal to $\frac{\varepsilon}{2}$ for each of the two phases of the hierarchical consensus method, in order to achieve the desired accuracy ε in the approximation of the global consensus state of the original graph G. Without a significant loss of generality, as it is detailed later in Section 4.4.3, the upper bound $t_{1^\circ\text{phase}}$ can be computed considering only the behavior of the graph with the largest μ_m;

- at time $t_{1^\circ\text{phase}}$, the auxiliary graph is considered;

- the matrix P_{aux} and the corresponding second-largest eigenvalue modulus μ_{aux} are computed, and an upper bound $t_{2^\circ\text{phase}}$ on the number of iterations needed to reach the required accuracy $\frac{\varepsilon}{2}$ in the approximation of its consensus state is determined, still with respect to the l_∞–norm. The vector of initial opinions of the agents associated with the nodes of the auxiliary graph is constructed in such a way that the consensus state of the auxiliary graph approximates the global consensus state of the graph G within the accuracy $\frac{\varepsilon}{2}$ (see Section 4.4.2 for the definition of such a vector of initial opinions).

Summarizing, the time needed by the first phase of the hierarchical con-

sensus method to terminate is equal to $t_{1^\circ\text{phase}}$, and depends mainly on the expression $\mu_{\max} := \max_{m=1,\dots,M}\{\mu_m\}$, whereas the time needed by the second phase to terminate is equal to $t_{2^\circ\text{phase}}$, and depends mainly on μ_{aux}. It follows that the proposed hierarchical consensus method has better performance guarantees than the non-hierarchical consensus method if the following condition is met:

$$t_{1^\circ\text{phase}} + t_{2^\circ\text{phase}} < T\,. \tag{4.5}$$

More details about this comparison are provided in Section 4.4.3.

4.4.2 Definitions of the vectors of initial opinions, and asymptotic analysis

In this subsection, we aim at studying the solution computed by the hierarchical consensus method when the numbers of iterations of both its phases are sufficiently large (ideally, when both $t_{1^\circ\text{phase}}$ and $t_{2^\circ\text{phase}}$ tend to infinity, or equivalently, when the tolerance ε tends to 0), to verify that it can really provide a good approximation of the global consensus state of the graph G, if the initial opinions of the nodes belonging to the subgraphs and to the auxiliary graph are chosen properly.

We recall that, in the first phase of the proposed hierarchical consensus method, one determines M connected subgraphs $G_m = (V_m, E_m)$, with $m = 1, \dots, M$. Here, we denote the number of nodes of each subgraph by $N_m = |V_m|$. Since the proposed procedure allows the presence of overlaps of nodes, i.e., it may happen that the same node is shared by different subgraphs, it is important to deal with such shared nodes properly (this issue is present only if the *nearest supernode approach* is applied in the first phase, since there are no shared nodes when the spectral clustering is applied). In the following, we suppose that, when evolving each subgraph, one knows which nodes are shared with other subgraphs, and the number of such node–sharing subgraphs (this is a mild assumption, since this information could be provided by the agent associated with the node itself). We define the vectors $x_m(0) \in \mathbb{R}^{N_m}$ ($m = 1, \dots, M$) of initial opinions of the agents belonging to the subgraphs in the following way.

If a component of $x_m(0)$ (say the p–th component, denoted by $x_m^{(p)}(0)$) refers to a node of G which is not shared with other subgraphs (say the node q), then we set

$$x_m^{(p)}(0) = x^{(q)}(0)\,, \tag{4.6}$$

where the right–hand side refers to the q–th component of $x(0)$. It the node p is shared, say, by M_p subgraphs, then we set

$$x_m^{(p)}(0) = \frac{x^{(q)}(0)}{M_p}\,. \tag{4.7}$$

In this way, the opinions of the agents associated with nodes shared by various subgraphs are rescaled. Now, at the end of the first phase, when $t_{1^\circ\text{phase}}$ is sufficiently large, one has, for all $m = 1, \ldots, M$, and for each i–th component of $x_m(t_{1^\circ\text{phase}})$,

$$x_m^{(i)}(t_{1^\circ\text{phase}}) \simeq \frac{1}{|V_m|}\mathbf{1}_m^T x_m(0)\,, \tag{4.8}$$

where $\mathbf{1}_m$ denotes a vector of all 1s, of the same dimension $|V_m|$ as $x_m(0)$, and formula (4.8) holds since (3.2) can be applied in the analysis. Hence, the local consensus state of each subgraph is equal to the average of the initial opinions of the agents associated to that subgraph, possibly rescaling the values of the opinions of the agents associated with nodes shared by different subgraphs (thus, the local consensus state of a subgraph that shares some nodes with other subgraphs may be different from its local consensus state in case of no shared nodes). Without loss of generality, in the following we assume that the supernodes are associated with $i = 1$ in formula (4.8).

At this point, at the beginning of the second phase of the hierarchical consensus method (i.e., at time $t = t_{1^\circ\text{phase}}$), we define the vector $x_{\text{aux}}(t_{1^\circ\text{phase}}) \in \mathbb{R}^M$ of initial opinions of the agents associated with the nodes of the auxiliary graph as follows:

$$x_{\text{aux}}(t_{1^\circ\text{phase}}) = \left[|V_1|\frac{M}{N}x_1^{(1)}(t_{1^\circ\text{phase}}), \ldots, |V_M|\frac{M}{N}x_M^{(1)}(t_{1^\circ\text{phase}})\right]^T, \tag{4.9}$$

i.e., the opinion of the agent associated with the m–th supernode is rescaled by the factor $|V_m|\frac{M}{N}$. Finally, by a similar analysis, the consensus state of the auxiliary graph (which is achieved within an arbitrary accuracy if $t_{2^\circ\,\mathrm{phase}}$ is sufficiently large) is the average of such opinions, and is equal to

$$\frac{\sum_{m=1}^{M}|V_m|\frac{M}{N}x_m^{(1)}(t_{1^\circ\mathrm{phase}})}{M} \simeq \frac{\sum_{m=1}^{M}|V_m|\frac{M}{N}\frac{1}{|V_m|}\mathbf{1}_m^T x_m(0)}{M} = \frac{1}{N}\mathbf{1}^T x(0)\,, \tag{4.10}$$

where the last expression is just the desired global consensus state of the original graph G.

4.4.3 Performance analysis

In this subsection, we provide a performance analysis of the proposed hierarchical consensus method, comparing it with the non–hierarchical consensus method, and expressing condition (4.5) in terms of spectral properties of the graphs/subgraphs involved in the method. A similar analysis was made in (97) for a similar hierarchical consensus method developed therein, but in that case, no spectral graph arguments were presented to motivate that method.

In the following, we investigate the number of iterations needed by the hierarchical consensus method to reach an approximation of the global consensus state up to the tolerance $\varepsilon > 0$. The following discussion refers to any among the graphs G, G_m, and G_{aux}, although we exemplify it at first by considering the graph G. Using formula (4.3), the minimum number of iterations of formula (3.1) that guarantees an approximation of the consensus state up to the desired tolerance $\varepsilon > 0$ is equal to

$$T = \max\left\{0, \frac{\log\left(\frac{\varepsilon^2}{|V|\|x(0)\|_\infty^2}\right)}{\log(\mu)}\right\}. \tag{4.11}$$

Since $\mu < 1$, one gets $\log(\mu) < 0$, while $\log\left(\frac{\varepsilon^2}{|V|\|x(0)\|_\infty^2}\right)$ could either be positive or negative. In particular, its numerator is positive when $\varepsilon > \sqrt{|V|}\|x(0)\|_\infty$, from which it follows $\frac{\log\left(\frac{\varepsilon^2}{|V|\|x(0)\|_\infty}\right)}{\log(\mu)} < 0$ and $T = 0$.

Moreover, for the case of a sufficiently small value of ε, one has $\varepsilon < \sqrt{|V|}\|x(0)\|_\infty$ (hence, a positive value for T), and formula (4.11) becomes

$$T = \frac{\log\left(\frac{\varepsilon^2}{|V|\|x(0)\|_\infty^2}\right)}{\log(\mu)}\,. \tag{4.12}$$

A similar kind of bound holds, with obvious changes in notations, for the subgraphs G_m and for the auxiliary graph $G_{\rm aux}$. In the following, we always assume that the associated ε is sufficiently small, in such a way that simplifications like (4.12) can be made. In particular, for the first phase of the hierarchical consensus method, one gets

$$t_{1^\circ{\rm phase}} \leq \max_{m=1,\ldots,M}\left\{\frac{\log\left(\frac{\varepsilon^2}{4|V_m|\|x_m(0)\|_\infty^2}\right)}{\log(\mu_m)}\right\}, \tag{4.13}$$

and, for its second phase,

$$t_{2^\circ{\rm phase}} \leq \left\{\frac{\log\left(\frac{\varepsilon^2}{4|V_{\rm aux}|\|x_{\rm aux}(t_{1^\circ{\rm phase}})\|_\infty^2}\right)}{\log(\mu_{\rm aux})}\right\}. \tag{4.14}$$

At this point, we observe that an advantage of using the l_∞–norm in the analysis (with respect, e.g., to the l_2–norm) is that, since the state vector in (3.1) is a convex combination of the opinions of the agents associated with the nodes of the graph, one gets

$$\|x(t)\|_\infty \leq \|x(0)\|_\infty\,, \forall t = 1, 2, \ldots\,. \tag{4.15}$$

This, combined with the definitions (4.6), (4.7), and (4.9), provides also the following upper bounds:

$$\|x_m(0)\|_\infty \leq \|x(0)\|_\infty\,, \forall m = 1, \ldots, N\,, \tag{4.16}$$

$$\|x_{\rm aux}(t_{1^\circ{\rm phase}})\|_\infty \leq \max_{m=1,\ldots,M}\{|V_m|\}\,\frac{M}{N}\|x(0)\|_\infty\,, \tag{4.17}$$

which allow one to bound from above $t_{1^\circ{\rm phase}}$ and $t_{2^\circ{\rm phase}}$ in terms of

$\|x(0)\|_\infty$ as follows:

$$t_{1^\circ\text{phase}} \leq \max_{m=1,\ldots,M} \left\{ \frac{\log\left(\frac{\varepsilon^2}{4|V_m|\left(\max_{m=1,\ldots,M}\{|V_m|\}\frac{M}{N}\right)^2 \|x(0)\|_\infty^2} \right)}{\log\left(\mu_m\right)} \right\}, \tag{4.18}$$

$$t_{2^\circ\text{phase}} \leq \left\{ \frac{\log\left(\frac{\varepsilon^2}{4|V_{\text{aux}}|\|x(0)\|_\infty^2} \right)}{\log\left(\mu_{\text{aux}}\right)} \right\}. \tag{4.19}$$

For ε sufficiently small (in particular, smaller than 1, in such a way that $\log(\varepsilon)$ is negative), the right–hand sides of formulas (4.12), (4.18), and (4.19) are dominated, respectively, by the terms $2\frac{\log(\varepsilon)}{\log(\mu)}$, $2\frac{\log(\varepsilon)}{\log(\mu_{\max})}$, and $2\frac{\log(\varepsilon)}{\log(\mu_{\text{aux}})}$. Then, the hierarchical consensus method has better performance guarantees than the non-hierarchical consensus method when the following condition holds:

$$\frac{1}{\log\left(\mu_{\max}\right)} + \frac{1}{\log\left(\mu_{\text{aux}}\right)} > \frac{1}{\log\left(\mu\right)}, \tag{4.20}$$

(here, one can notice that all the ratios involved are negative).

Concluding, formula (4.20) shows that the hierarchical consensus method is associated with better performance guarantees than the non–hierarchical one when all the subgraphs G_m ($m = 1, \ldots, M$) and the graph G_{aux} have better spectral properties than G.

4.5 Numerical examples and results

In the following, we present some numerical examples of the suggested procedure. In particular we perform the $1st$ step of the hierarchical procedure by applying both the spectral clustering algorithm and the *nearest supernode approach* with the aim of comparing the two methods. In both the situations we evaluate the results changing the number M of subgraphs extracted from the original graph G. We test the procedure with random graphs; first we apply the method to a random geometric graph whose construction is explained later, then we apply the

method to a planted partition model (101; 102) and finally to a preferential attachment model (103). This model generates random scale–free networks (104; 105; 106) such the Internet network, the World Wide Web, citation networks, and some social networks. In particular, for all the types of graphs considered, we test both spectral clustering and the *nearest supernode approach* on graphs with a relative small number of nodes, such as 100 and 300, in order to reduce the computational effort especially when spectral clustering algorithm is applied. In addition, when a planted partition graph is considered, two examples are considered, for different choices of the intra– and inter–clusters probability values (parameters that will be better explained in this chapter). The idea is to test the performances of both the methods with examples where clusters are more visible and easier to detect and with examples where the cluster–exhibiting structure is less defined and clear. When applying the *nearest supernode approach*, we first fix the number M of subgraphs, then we run 10 tests for each situation studied. In fact, in the process of computing the subgraphs, the nodes in ON are randomly picked up and then assigned to the *supernodes*. Thus, we run different tests in order to obtain statistically significant results and better comparisons. The final result reported is then the mean and the standard deviation over the 10 tests. For every kind of graph considered in the numerical comparison, the vector $x(0)$ of the initial opinions of the N agents is generated as the realization of a random vector, where each component is drawn i.i.d. from the standard uniform distribution on the interval $(0, 1)$. The l_∞–norm of this vector is then used to determine the minimal number of steps of the non–hierarchical consensus method that guarantees to reach the global consensus state up to the fixed tolerance $\varepsilon > 0$ (see Formula (4.11)). It is worth noting that, with this choice of the vector $x(0)$, one can bound from above its l_∞–norm by the value 1, without knowing the specific realizations of its components. Formulas (4.18) and (4.19) are then used for the two phases of the hierarchical consensus method. Finally, we consider values of ε sufficiently small in order to neglect the dependence of formulas (4.12), (4.18), and (4.19) on the number of nodes of the subgraphs/graphs considered, and to assume that the slowest subgraph in

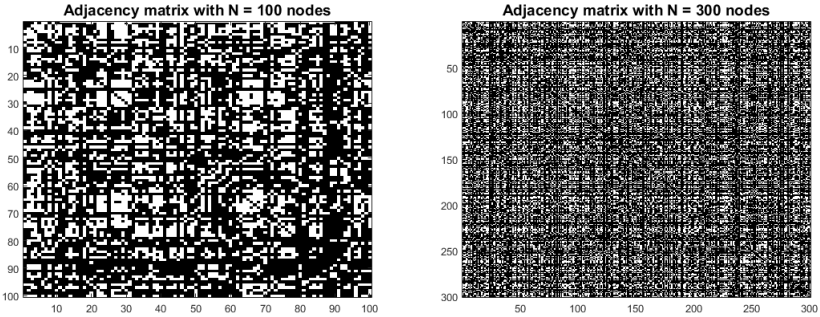

Figure 10: Adjacency matrices of a random geometric graph. On the left an example with 100 nodes, on the right an example of graph with 300 nodes.

the first phase of the consensus method is the one associated with $\mu_{\max}$.

4.5.1 Random geometric graph

To generate this kind of graph, we sample N points from a 3 dimensional Gaussian distribution with mean $(0, 0, 0)$ and covariance matrix = $3I_3$. A threshold is then applied on the distances between every pair of points connecting with an edge the points with a distance smaller than the threshold. Two random geometric graphs with different dimension are considered. The first one with $N = 100$ nodes and the second one with $N = 300$ nodes. The two adjacency matrices are shown in Figure 10. In particular, in both the examples, we fix a tolerance ε equal to 10^{-6}. We estimate the number of steps required by the algorithm by considering Formula (4.11) with the infinity norm.

Random geometric graph with $N = 100$ nodes

In this case study, the second–largest eigenvalue modulus of the transition probability matrix P associated to this graph is equal to $\mu = 0.97$, while the number of steps required by the classical algorithm applied directly to the original graph up to the tolerance ε are $T = 458$. We compute the $1st$ phase of the hierarchical method by considering both

the spectral clustering algorithm and the *nearest supernode*. In particular, when the spectral clustering method is applied we require the method to extract a number of clusters $M \in \{10, 5, 2, 1\}$, while with the *nearest supernode approach* we compute the results when $M \in \{20, 10, 5, 2, 1\}$. We do not require the spectral clustering algorithm to determine 20 clusters because that algorithm could create disconnected subgraphs. Clearly, the result obtained for $M = 1$ is the one achieved by the consensus algorithm applied to the original graph G. In the figures we report the results associated to $M = 1$ in order to have a clear comparison between the time of convergence to the consensus state of the consensus algorithm applied to G and the sum of the times of the $1st$ and $2nd$ steps of the hierarchical method.

Figure 11 reports $t_{1^\circ \text{step}} + t_{2^\circ \text{step}}$, i.e., the sum of the time needed by the slowest subgraph to reach its own consensus during the $1st$ phase and the time needed by the auxiliary graph to reach the final result, varying the number M of subgraphs considered. On the left it is shown the time needed by the method when spectral clustering is applied during the $1st$ phase of the technique, while on the right the results obtained by the *nearest supernode approach*. For both the figures, the one on the left that is obtained by applying the clustering algorithm for the $1st$ phase and the one on the right that shows the results when the *nearest supernode approach* is applied, we report on the x–axis an indicative value of the dimension of the subgraphs. In fact in both the situations we do not know the exact dimension of each subgraph. When the spectral clustering algorithm is applied, the method requires in input the number of clusters M one wants to detect and not their dimension, hence the dimensions of the subgraphs change; while with the *nearest supernode approach* overlaps of nodes are allowed among the different M subgraphs due to the procedure followed for their construction, and again subgraphs with different dimensions are created. We indicate with $\frac{N}{M}$(C)/ $\frac{N}{M}$(K) the indicative dimension of the subgraphs when the spectral clustering/*nearest supernode approach*, respectively, is considered. Note that, when the *nearest supernode approach* is considered, in case of no overlaps the average dimension of the subgraphs takes a value $k = \frac{N}{M}$, while since in our situation over-

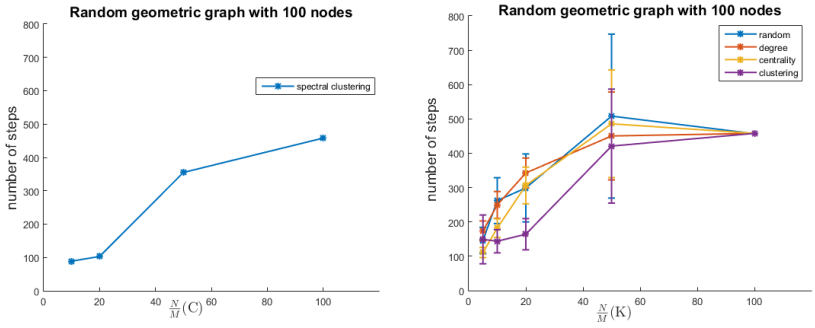

Figure 11: Number of steps to reach the consensus state with a random geometric graph with 100 nodes. On the left: spectral clustering is used during the $1st$ phase of the algorithm; on the right: the $1st$ phase of the algorithm is computed by applying the *nearest supernode approach* selecting the *supernodes* with the 4 different types of seeds.

laps are allowed, in general the dimension of the subgraphs G_m is greater than $k = \frac{N}{M}$. However, the *nearest supernode approach* creates subgraphs even with a number of nodes smaller than k. The plot shows the mean and standard deviation of the number of steps required by the hierarchical procedure over the 10 tests when *nearest supernode approach* is used in the $1st$ phase of the method. The four types of seeds are considered.

Random geometric graph with $N = 300$ nodes

Concerning the random geometric graph with $N = 300$ nodes, whose adjacency matrix is the one on the right in Figure 10, one obtains $\mu = 0.995$. The number of steps required by the algorithm to reach the consensus state up to a tolerance ε are $T = 2543$. On the left of Figure 12 the results obtained when spectral clustering is applied to compute the $1st$ phase of the method are shown. In particular, a number of clusters $M \in \{30, 15, 6, 3, 2, 1\}$ are required to be detected. Again we do not consider a higher number of clusters, such as 60, because the algorithm has problem in detecting connected subgraphs. When the *nearest supernode approach* is considered, the number of subgraphs used to divide the orig-

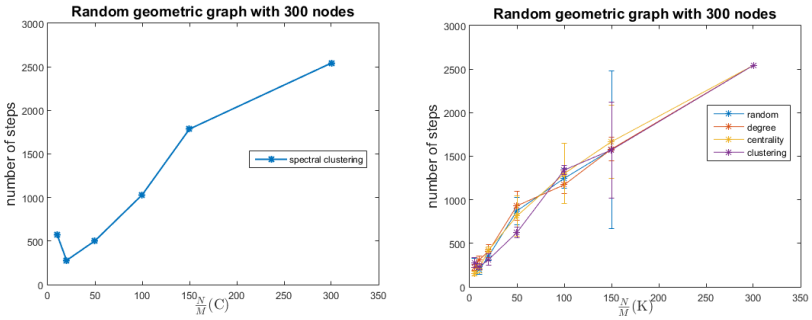

Figure 12: Number of steps to reach the consensus state with a random geometric graph with 300 nodes. On the left: spectral clustering is used during the $1st$ phase of the algorithm; on the right: the $1st$ phase of the algorithm is computed by applying the *nearest supernodeapproach* selecting the *supernodes* with the 4 different types of seeds.

inal graph, instead, are $M \in \{60, 30, 15, 6, 3, 2, 1\}$. The result is the plot on the right of Figure 12 and it is obtained by averaging over 10 tests the result of each run. The results corresponding to $M = 1$ are the ones obtained by the consensus algorithm directly applied to the original graph G.

From the plot shown in Figure 12, we can infer that for each type of seed used and for each value of k, when random geometric graphs are considered, the hierarchical method improves the results obtained by the classical algorithm. In addition, when a random geometric graph with a relatively high number of nodes (300 nodes rather then 100 nodes) is considered, the *nearest supernode approach* works even better than the spectral clustering algorithm. When small subgraphs are generated, with an approximate number of nodes equal to 5 or 10, and even when the original graph is divided in only $M = 2$ subgraphs, the consensus state is reached faster applying the *nearest supernode approach* for the $1st$ phase.

4.5.2 Planted partition graph

The same procedure is then applied to a planted partition model. This kind of graph is a cluster–exhibiting random graph model, where

nodes inside the same cluster are connected by an edge with probability p, while nodes belonging to different clusters are connected with probability equal to q (107; 108). In particular, we follow the Erdos–Renyi model that generates a random graph with N nodes, that exhibits two clusters: the first one with dimension N_1 and the second one with N_2 nodes. In the following we will indicate with p_{in} the probability of interconnection p inside each single cluster, while p_{out} stands for the probability q of interconnections among nodes belonging to different clusters. We consider examples with two equal–sized clusters. Thus, starting from a graph with N nodes, we require each cluster to have dimension equal to $\frac{N}{2}$.

Planted partition graphs with $N = 100$ nodes

We first consider two graphs with $N = 100$ nodes but different values for the probabilities p_{in} and p_{out}. The first one, whose adjacency matrix is shown on the left of Figure 13, has intra–cluster probability of connection equal to $p_{in} = 0.1$, while points in different clusters are connected with probability $p_{out} = 0.02$. On the right of Figure 13, instead, the graph shown has probability of intra–cluster connection equal to 0.2, while points in different clusters are connected with probability 0.01. We refer to the last graph as the denser graph between the two, while we indicate the first one as the sparser one. We first study the sparser case (adjacency matrix on the left of Figure 13). The number of steps required by the algorithm to reach the consensus state up to a tolerance $\varepsilon = 10^{-6}$ are $T = 239$, while the second–largest eigenvalue modulus takes a value $\mu = 0.94$. The results obtained by the hierarchical method by considering spectral clustering and the *nearest supernode approach* to complete the $1st$ phase are shown in Figure 14.

Concerning the denser graph (adjacency matrix on the right of Figure 13), the results are shown in Figure 15. In particular, on the original graph, the number of steps required to reach an approximation of the consensus state up to a tolerance equal to $\varepsilon = 10^{-6}$ are $T = 326$ while the second–largest eigenvalue modulus is equal to $\mu = 0.96$.

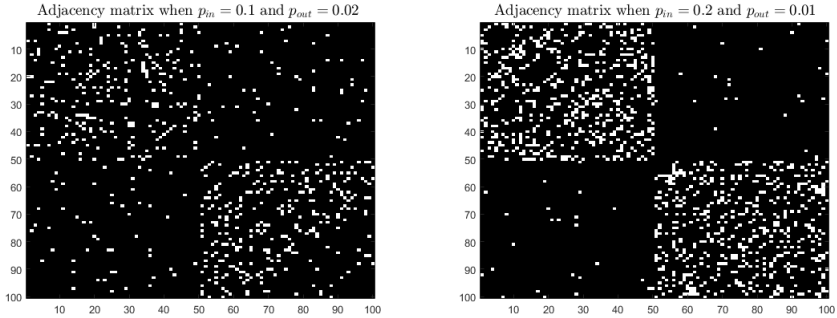

Figure 13: Adjacency matrices of a planted partition graph with 100 nodes. On the left an example with $p_{in} = 0.1$ and $p_{out} = 0.02$, on the right an example with $p_{in} = 0.2$ and $p_{out} = 0.01$.

When dealing with planted partition graphs, the *nearest supernode approach* does not work as well as in the case with random geometric graphs. In particular, when sparser graphs are considered, this method is not able to outperform the result achieved by the consensus algorithm directly applied to the original graph, as Figure 14 shows. Nevertheless, if the $1st$ phase is performed by the spectral clustering algorithm, we obtain satisfactory results (plot on the left of Figure 14). In fact, all the choices for the number of clusters considered to partition the original graph lead to an increase of the convergence rate to the consensus state. As expected, when two clusters are required, the method obtains the best result. When denser subgraphs are considered, instead, even the *nearest supernode approach* is able to achieve good results, as Figure 15 shows, especially when the clustering coefficient is adopted to select the *supernodes* and subgraphs with a small number of nodes (approximately equal to 5 or 10) are considered.

Planted partition graph with $N = 300$ nodes

Finally, we report another example that considers a planted partition graph with $N = 300$, $p_{in} = 0.2$ and $p_{out} = 0.01$; the corresponding adjacency matrix is shown in Figure 16. The number of steps required by

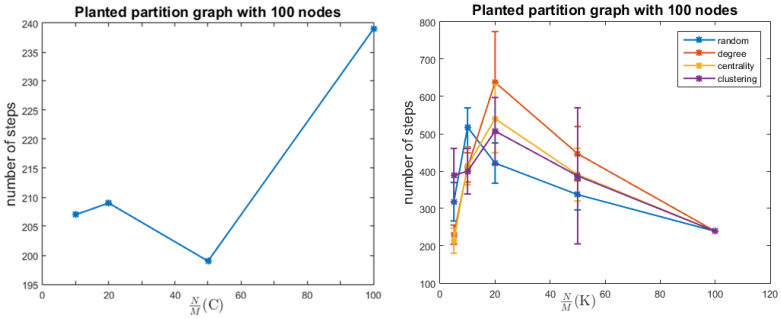

Figure 14: Number of steps to reach the consensus state with a planted partition graph with 100 nodes with $p_{in} = 0.1$ and $p_{out} = 0.02$. On the left: spectral clustering for the $1st$ phase of the algorithm; on the right: *nearest supernode approach*.

the consensus algorithm to reach the consensus state up to a tolerance equal to 10^{-6} are $T = 213$, while the second–largest eigenvalue modulus is equal to $\mu = 0.94$. The results achieved by the spectral clustering algorithm and by the *nearest supernode approach* are shown on the left of Figure 17 and on the right of Figure 17, respectively. Note that when spectral clustering is adopted, the original graph is divided in a number of subgraphs $M \in \{30, 15, 6, 3, 2, 1\}$; while, wen we compute the $1st$ step of our method by means of the *nearest supernode approach*, we divide the original graph in a number of subgraphs $M \in \{60, 30, 15, 6, 3, 2, 1\}$. Again, we can infer that the spectral clustering method works better than the *nearest supernode approach*, even if the latter is able to achieve good results when subgraphs G_m with approximately a small number of nodes, i.e., 5 or 10, are extracted from graph G. A detailed analysis of the behavior of the *nearest supernode approach* will be provided in Section 4.6.

4.5.3 Preferential Attachment model

We conclude the study of the hierarchical method by applying it to a Preferential Attachment (PA) model. The graph is generated according to the $G(N, m)$ model (109), where the number of edges inserted whenever a new node is added is $m = 2$, while for N, we again choose to

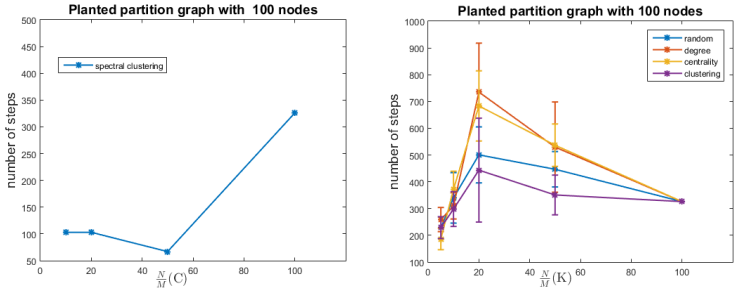

Figure 15: Number of steps to reach the consensus state with a planted partition graph with 100 nodes with $p_{in} = 0.2$ and $p_{out} = 0.01$. On the left: spectral clustering for the $1st$ phase of the algorithm; on the right: *nearest supernode approach* for the $1st$ phase.

test the method to both a graph with $N = 100$ nodes (adjacency matrix shown on the left of Figure 18) and one with 300 nodes, whose adjacency matrix is the one on the right of Figure 18.

PA model with $N = 100$ nodes

The second–largest eigenvalue modulus of the transition probability matrix associated to the original graph is $\mu = 0.99$ while the number of steps required by the algorithm to reach the consensus state up to a tolerance equal to $\varepsilon = 10^{-6}$ are $T = 937$. The results obtained by applying the hierarchical method are shown in Figure 19.

PA model with $N = 300$ nodes

Finally, we perform the same study with a PA model with $N = 300$ nodes; the adjacency matrix is shown on the right of Figure 18. For this example, the second–largest eigenvalue modulus of the transition probability matrix takes a value $\mu = 0.99$, while the number of steps needed by the consensus algorithm directly applied to the original graph to reach an approximation of the consensus state equal to $\varepsilon = 10^{-6}$ are $T = 1005$. The results obtained by applying both the spectral clustering algorithm

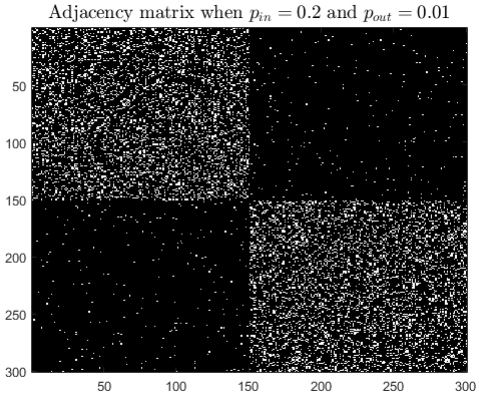

Figure 16: Adjacency matrix of a planted partition graph with 300 nodes, $p_{in} = 0.2$ and $p_{out} = 0.01$.

and the *nearest supernode approach* are shown in Figure 20.

With a PA model the *hierarchical method* shows some problems with both the methods adopted during the $1st$ phase, especially with quite large graphs (worse results are achieved with the graph with 300 nodes). Nevertheless, when smaller graphs G are considered, i.e., the case with 100 nodes, good results are obtained. In particular, the *nearest supernode approach* with subgraphs G_m, $m = 1, \ldots, M$ with a relatively small number of nodes is able to increase the convergence rate to the consensus state. In addition, as the plot on the right in Figure 19 shows, the best results are obtained when the clustering coefficient is used as seed for the selection of the *supernodes*.

4.6 Drawbacks and refinements of the basic version of the method

In the previous sections, we have applied the hierarchical consensus method to different kinds of graphs, using two clustering methods for the extraction of the subgraphs G_m. We have observed that both methods are able to achieve satisfactory results when realizations of random geometric graphs are considered. Indeed, in this case, better results have

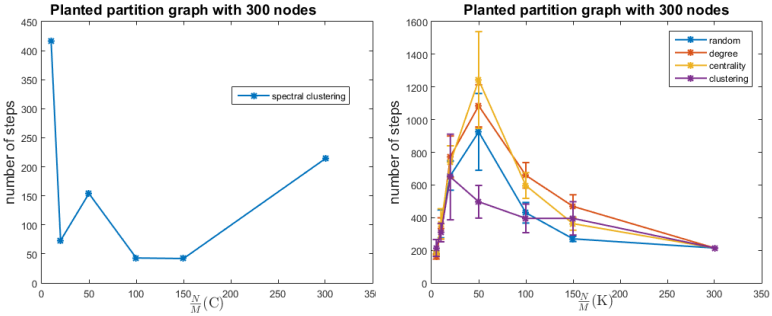

Figure 17: Number of steps to reach the consensus state with a planted partition graph with 300 nodes with $p_{in} = 0.2$ and $p_{out} = 0.01$. On the left: spectral clustering for the $1st$ phase of the algorithm; on the right: *nearest supernode approach* for the $1st$ phase.

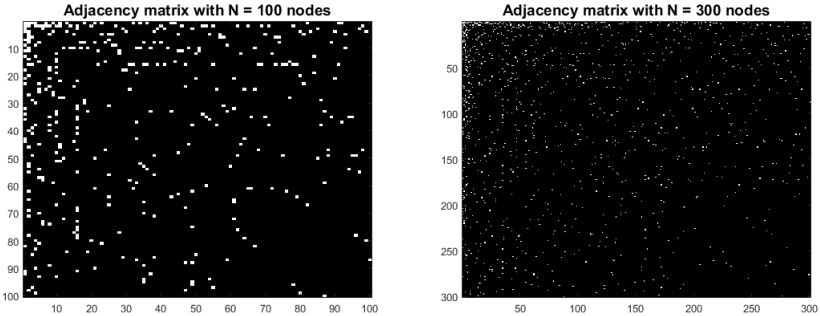

Figure 18: Adjacency matrices of a PA model. On the left an example with 100 nodes, on the right an example with 300 nodes.

been always obtained than the ones achieved by the non–hierarchical consensus method, as Figures 11 and 12 show. When realizations of planted partition models are considered, the results highlight the fact that, when a cluster–exhibiting graph G is considered, the hierarchical consensus method implemented via spectral clustering has better performance than the non–hierarchical consensus method; while the *nearest supernode approach* reveals some drawbacks. Finally, when considering realizations of the preferential attachment model, both clustering methods show problems. Nevertheless, the *nearest supernode approach* achieves better results than spectral clustering, and in some situations it is able to

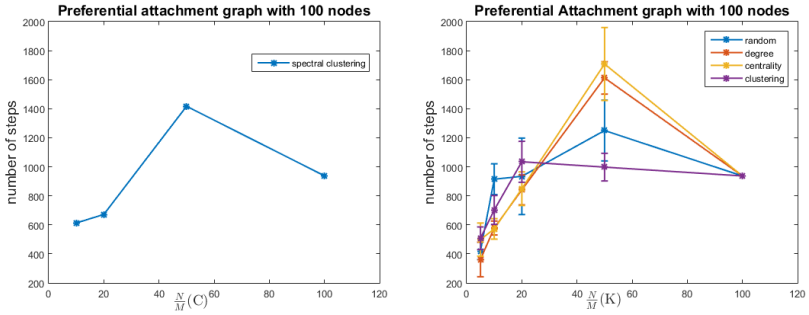

Figure 19: Number of steps to reach the consensus state with a PA model with 100 nodes. On the left: spectral clustering for the $1st$ phase of the algorithm; on the right: *nearest supernode approach* for the $1st$ phase of the method.

outperform the non-hierarchical consensus method.

In this section, first, we analyze a factor, which we will call *antenna effect*, that is shown to influence strongly (and negatively) the results achieved by both clustering methods adopted to perform the first phase of the hierarchical consensus method. Subsequently, we provide a possible way to overcome that effect. Finally, we discuss the choice of the *supernodes* in the *nearest supernode approach*.

4.6.1 The *antenna effect*

In this subsection, we analyze the performance of the hierarchical consensus method when subgraphs G_m containing one node with degree equal to 1 are generated. We refer to this kind of situation by the term *antenna effect*. To simplify the theoretical analysis, we consider here the case of a graph like the one shown in Figure 21, which is made of a complete subgraph (in this case, made of $N - 1 = 4$ nodes) connected by a single edge to another node with degree equal to 1. We call this kind of graph *basic antenna effect model* with N nodes. In the next subsection, we also investigate numerically other kinds of graphs showing the occurrence of the antenna effect.

We aim at studying this kind of situation making use of the Cheeger's

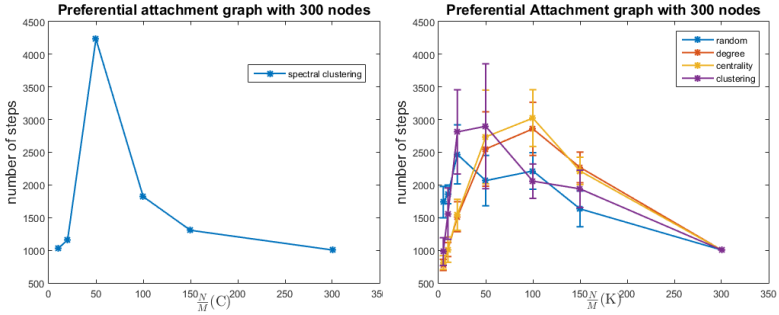

Figure 20: Number of steps to reach the consensus state with a PA model with 300 nodes. On the left: spectral clustering for the $1st$ phase of the algorithm; on the right: *nearest supernode approach* for the $1st$ phase of the method.

inequality, see Section 2.3.3. We briefly recall that this inequality provides a lower and an upper bound on the second–smallest eigenvalue of the normalized Laplacian of the graph considered (see, e.g., Formula (2.7)) by means of the Cheeger's constant (Formula (2.6)). Given a graph $G = (V, E)$, the Cheeger's constant is computed by considering all the subgraphs $S \subset V$ that partition G in two subgraphs S and $V \setminus S$. For our investigation of the *antenna effect*, we do not need to compute exactly the Cheeger's constant appering inside the Cheeger's inequality (which is a combinatorial problem, see Formula (2.7)), but we limit to find an upper bound on it.

Now, in our particular context, we can see the transition probability matrix P as the weighted adjacency matrix A of the graph G. We recall that P, hence A, in the average consensus problem is a doubly stochastic matrix; thus the normalized Laplacian matrix $L_N = I - A = I - P$, since the degree matrix D is exactly equal to the identity matrix I. It follows that the eigenvalues of the Laplacian matrix ξ_i and the eigenvalues of the transition probability/adjacency matrix λ_i are related by the following relation:

$$\xi_i = 1 - \lambda_i, \text{ for } i = 0, \ldots, N-1. \tag{4.21}$$

We know that the convergence rate to the consensus state depends on

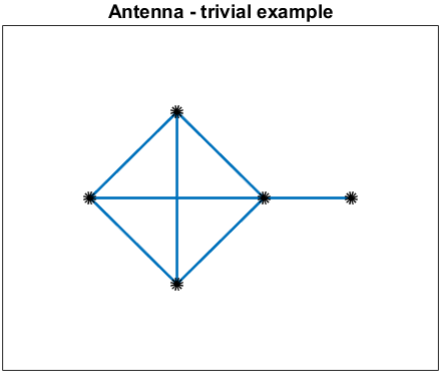

Figure 21: A nearly complete graph with a node attached to only one node of the complete part.

the value of the second–largest eigenvalue modulus of P, μ; thus, given Formula (4.21), it follows that the larger the second–smallest eigenvalue ξ_1 of L_N, the faster the convergence rate to the consensus state. Note that these considerations hold when $\xi_1 < 1$ (otherwise it would be necessary to consider ξ_{N-1}); anyway, as (71) shows, it is possible to modify the original graph by adding self–loops with a certain value to every vertex of the graph in order to have $\xi_1 < 1$ without changing the results and the particular situation studied.

Now, we aim at at studying how the eigenvalue ξ_1 is influenced by the occurrence of the *antenna effect*. To do this, we exploit the Cheeger's inequality to find an upper bound on ξ_1 for the basic *antenna effect* model with $N \geq 2$ nodes.

We recall that the matrix P, which from now on is treated as the weighted adjacency matrix A of graph G, is computed following the procedure described in Section 4.2.1. Thus, to each edge of G one associates in P a weight w (ϵ according to the notation used in Section 4.2.1), while a self-loop with weight $1 - wd_i$ is associated in P to every vertex i, where d_i is the corresponding degree. Now, for the basic *antenna effect* model, the largest degree $d_{\max}(G)$ in the graph is achieved by the only node of

the complete part of the graph which is connected to the node of degree 1, and is equal to $N-1$. Hence, since the weight of each self-loop has to be non-negative and smaller than or equal to 1, one obtains the bounds

$$0 \leq w \leq \frac{1}{N-1}\,. \tag{4.22}$$

Moreover, choosing the set S in the definition of the Cheeger's constant as in Figure 22 and using formula (2.6), one gets

$$\Phi(P) \leq \frac{w}{\min\{1, N-1\}} = w\,. \tag{4.23}$$

This, combined with Formulas (2.7) and (4.22), provides the following upper bound on ξ_1 for the basic antenna effect model with $N \geq 2$ nodes:

$$\xi_1 \leq 2w \leq \frac{2}{N-1}\,. \tag{4.24}$$

Hence, we can conclude that the basic antenna effect model with $N \geq 2$ nodes has a very small value of ξ_1 , hence, also its rate of convergence to the (local) consensus state is small. It is also worth mentioning, instead, that, for $N \geq 3$, the complete subgraph with $N-1$ nodes inside the basic antenna effect model (i.e., the subgraph obtained disconnecting the node with degree 1, and replacing the weight $1-(N-1)w$ of the self-loop of the attached node with $1-(N-2)w$) has[1]

$$\xi_1 = (N-2)w \cdot \frac{N-1}{N-2} = (N-1)w\,, \tag{4.25}$$

whose maximum value is

$$\xi_1 = \frac{N-1}{N-2} \tag{4.26}$$

when w achieves its maximum admissible value $\frac{1}{N-2}$. When N is large, formula (4.26) simplifies to

$$\xi_1 \simeq 1\,. \tag{4.27}$$

Hence, we can conclude that the presence of the additional node in the basic antenna effect model can decrease significantly the value of the second–smallest eigenvalue of the normalized Laplacian matrix.

[1]Formula (4.25) is provided, e.g., in (71, Lemma 1.7) for the case $w = \frac{1}{N-2}$ (no self-loops), whereas its extension to the presence of self-loops is straightforward.

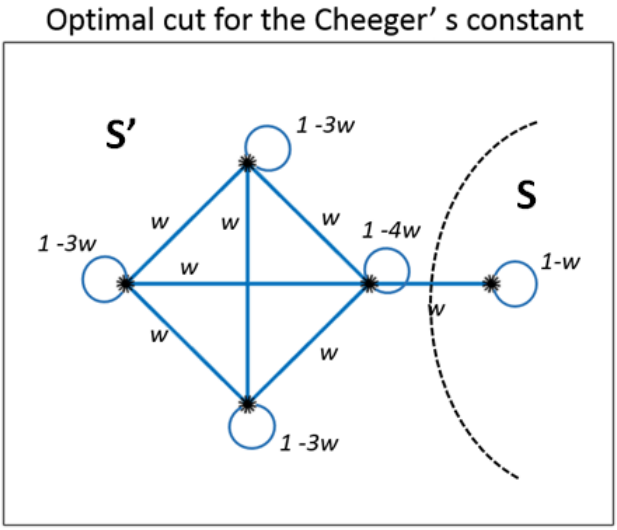

Figure 22: Choice of the subset S to determine an upper bound on the Cheeger's constant for the basis *antenna effect* model.

From the analysis presented above, we can conclude that, in situations for which the first phase of the hierarchical method can produce subgraphs showing the antenna effect, it is better to keep the average number of nodes of such subgraphs small. This explains why, in the numerical results presented in Section 4.5, good results have been obtained several times when, for instance, subgraphs with a small approximate average number of nodes $h = \frac{N}{M}$ (i.e., either 5 or 10), have been considered.

To support the theoretical analysis just presented, in the next subsection, we also investigate from a numerical point how the spectral properties of a graph can be influenced by the antenna effect.

4.6.2 Numerical examples related to the *antenna effect*

In the following, we examine two trivial examples of graphs presenting the *antenna effect*. Their adjacency matrices are shown in Figure 23; in particular, on the left we have a graph made of a complete

graph with 10 nodes with an additional node connected by means of only one edge; while the plot on the right shows the adjacency matrix of a random geometric graph (sparser than the complete one) with 50 nodes in total, where one of them has degree equal to 1. We perform the

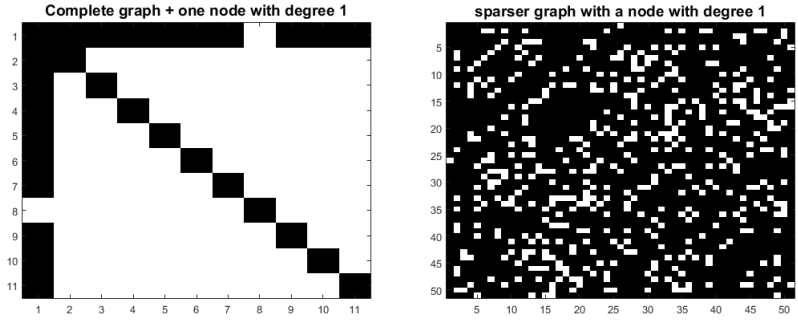

Figure 23: A complete graph on the left and a sparser one, on the right, with a node attached to only one node of the original graph.

following experiment. First, we consider the original graph (either the complete one with 10 nodes, or the sparser one with 50 nodes), and we compute the second–largest eigenvalue modulus of its associated transition probability matrix P. Then we connect the additional node to one selected node of the original graph, and we compute the second–largest eigenvalue modulus of the transition probability matrix P associated with the resulting graph. We repeat this procedure selecting each time a different node of the original graph, then we compare the resulting second–largest eigenvalue moduli. To do the comparison, we compute the transition probability matrix P in two ways: using the method described in Section 4.2.1, and also solving the Fastest Mixing Markov–Chain (FMMC) problem (48), which determines the optimal (i.e., smallest) value for the second–largest eigenvalue modulus when considering the non–hierarchical case. In this way, we avoid the possibility that an increase of the second–largest eigenvalue modulus obtained after the insertion of additional node has to be ascribed to the particular method adopted to determine the transition probability matrix P.

Concerning the complete graph, the second largest eigenvalue modulus obtained before the insertion of the additional node is approximately equal to 0, for both the methods adopted to compute the matrix P. When the additional node is inserted, connecting it every time with a different node of the original graph, one obtains a remarkable increase of the value of the second–largest eigenvalue modulus. Figure 24 shows the results: on the left the second–largest eigenvalue modulus of the matrix P computed with the method described in Section 4.2.1, on the right the results obtained by applying the FMMC method. The latter produces better results in terms of the value in modulus of the second–largest eigenvalue but the *antenna effect* still substantially worsen the final solution. Due to the nature of the original graph, that is a complete one, it was expected that the results achieved when one node with degree 1 is added do not depend on the choice of the node linked to the node added, as Figure 24 shows.

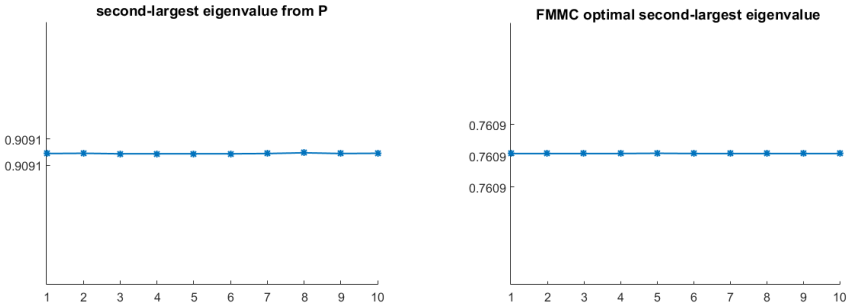

Figure 24: Second–largest eigenvalue modulus when a complete graph with the *antenna effect* is considered.

Regarding the second example (the one reported on the right of Figure 23), the results obtained by applying the same procedure are shown in Figure 25. For this example, the second–largest eigenvalue modulus of matrix P of the original graph computed with the method described in Section 4.2.1, is equal to $\mu = 0.8275$. The plot on the left of Figure 25 shows the increase of the second–largest eigenvalue modulus of P associated to the original graph adding every time a node connected to

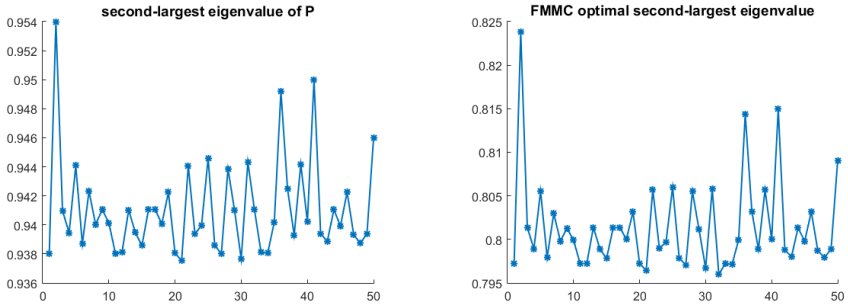

Figure 25: Second–largest eigenvalue modulus when a sparser graph than the complete one presents the *antenna effect*.

only one node of the original graph. On the other hand, if we apply the FMMC algorithm to the original graph we obtain a value of μ equal to 0.5606, while the increase of the second–largest eigenvalue modulus is shown in the plot on the right of Figure 25. Again, we can see that the results are remarkably worse with both the methods.

As an additional study, we analyze, from a numerical point of view, the effect of adding a pair of nodes with degree equal to one. In particular, we consider again the graph shown on the right of Figure 23, adding, each time, two nodes with degree equal to one instead of only one. Similarly to the previous experiment, we connect two nodes with degree equal to one to a specific selected node of the original graph, and we compute the second–largest eigenvalue modulus of the transition probability matrix P associated with the resulting graph. We repeat the procedure selecting each time a different node of the original graph. In this case, the value of the second–largest eigenvalue modulus of the transition probability matrix slightly increases. In fact, when the matrix P is computed by the method described in Section 4.2.1, considering 50 different combinations of pairs of nodes with degree equal to one, the average of the value of the second–largest eigenvalue modulus over the 50 tests is 0.95, wheres the standard deviation is 0.05. On the other hand, if the FMMC algorithm is applied to compute the second–largest eigenvalue modulus of the 50 different cases, one obtains 0.89 ± 0.02. It is interesting to

notice the increase of the second–largest eigenvalue modulus especially when the FMMC algorithm is applied. On the other hand, if two nodes with degree equal to one are connected to two different nodes and not only one, the average value of the second–largest eigenvalue modulus equal to 0.94 (standard deviation 0.04) is obtained, when the matrix P is computed by the method described in Section 4.2.1. When the FMMC algorithm is applied, instead, one obtains 0.81 ± 0.01.

Finally, we report the result obtained considering a planted partition graph with $N = 100$ nodes, $p_{in} = 0.2$ and $p_{out} = 0.01$. When dealing with this type of model, we use the term cluster to refer to the clusters generated by the model, while with subgraph we indicate the one extracted by the *nearest supernode approach*. In particular, for the following study, we apply the *nearest supernode approach* by selecting $M = 2$ subgraphs and the clustering coefficient as the seed to generate the *supernodes*. The adjacency matrix is shown in Figure 26; the second–largest eigenvalue modulus of the transition probability matrix associated to it and computed following the procedure in Section 4.2.1 is $\mu = 0.953$.

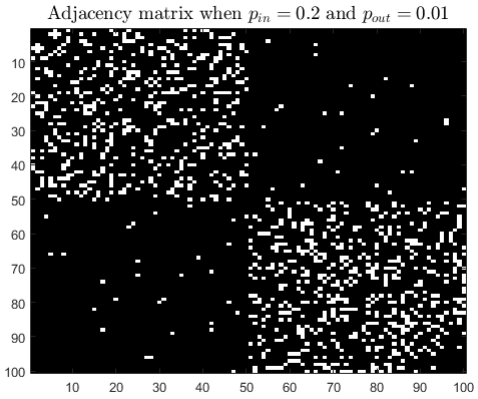

Figure 26: Adjacency matrix of a planted partition graph with 100 nodes, $p_{in} = 0.2$ and $p_{out} = 0.01$.

In Figure 27 the two subgraphs extracted from the original graph are shown. The second–largest eigenvalue modulus of the transition proba-

bility matrix associated to the first subgraph is equal to $\mu_1 = 0.854$, while the one associated to the second subgraph is $\mu_2 = 0.95$. Only one node wrongly assigned (the one in the second subgraph that originally belongs to the first cluster of the planted partition graph but it is wrongly assigned to the second cluster of the model) with degree 1 in the subgraph it is assigned to, considerably worsen the result.

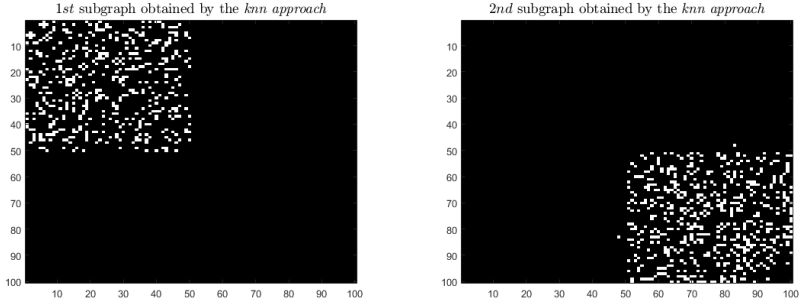

Figure 27: Subgraphs determined by the hierarchical method with clustering coefficient as seed. In the adjacency matrix on the right is shown the *antenna effect*.

The numerical examples just introduced provide an additional demonstration regarding the possibly high change of the spectral properties of a graph if one node with degree 1 is added. Thus, it is necessary to determine an automatic method able to avoid this situation. A possible solution is explained and presented later.

4.6.3 A solution to overcome the *antenna effect*

In this section we present a solution able to improve the results of the hierarchical method when subgraphs presenting the *antenna effect* are extracted from the original graph. In particular we test the method with the *nearest supernode approach*. The idea is to reassign the nodes with degree equal to 1 to subgraphs where their degree should be higher. More precisely, once the M subgraphs G_m with $m = 1, \ldots, M$ have been determined, we compute the degree of the nodes inside every single sub-

graph; if a node has degree equal to 1 in the subgraph it is assigned to, we compute its degree as if it would be assigned to all the other subgraphs and we finally assign it to the subgraph where its degree is maximum. In this way we try to avoid nodes with degree equal to 1 in the subgraphs they are assigned during the $1st$ phase of the method. To investigate numerically the effectiveness of the proposed solution to overcome the *antenna effect*, we apply it to a realization of the planted partition model with $N = 100$, $p_{in} = 0.1$ and $p_{out} = 0.02$, whose results are shown in the plot on the right of Figure 14. The results achieved are shown in Figure 28. The method is applied to the *nearest supernode approach* by considering all the four types of seeds. In blue the original results, and in red the ones obtained reassigning to other subgraphs the nodes with degree equal to 1 in the subgraph they are assigned at the beginning. The method is then applied to the realization of the planted partition graph with 300 nodes, $p_{in} = 0.2$ and $p_{out} = 0.01$ whose results are shown in Figure 17. The new results are reported in Figure 29. The results shown in the figures highlight that the procedure of reassigning the nodes with degree 1 in the subgraph they are assigned at the beginning remarkably improves the results. In the examples reported, the number of steps considerably decreases when all the types of seeds are considered. In particular, the best results are obtained when the clustering coefficient is chosen as seed to determine the *supernodes*. When a planted partition graph with a relatively high value for p_{in} is considered, like the one shown in Figure 16 satisfactory results are achieved, especially for subgraphs G_m with a relatively small approximate number of nodes, i.e. 5 or 10. When a graph with less intra–cluster connections is considered (the one reported on the left of Figure 13) it is more difficult to obtain better results than the one achieved by the consensus algorithm directly applied to the original graph but, remarkably improvements are still obtained and, with additional studies, even better results are expected.

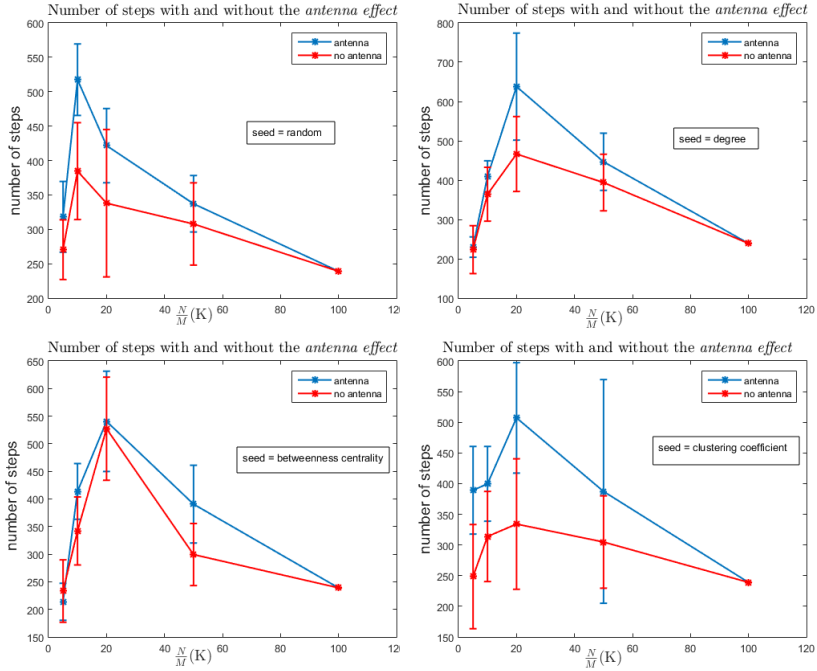

Figure 28: Planted partition graph with 100 nodes, $p_{in} = 0.1$ and $p_{out} = 0.02$. In blue: number of steps required by the original hierarchical method; in red: number of steps reassigning the nodes with degree 1 to other subgraphs.

4.6.4 Choice of the *supernodes* in the planted partition model

We conclude by mentioning another possible issue that can decrease the performance of the hierarchical consensus method, when the *nearest supernode approach* is applied during its first phase. This issue is important especially for planted partition models. The problem regards the choice of the *supernodes*: as it is shown in the following, bad performances are expected if the *supernodes* belong to the same "apparent cluster". Let us consider again the planted partition model with $N = 100$ nodes, $p_{in} = 0.2$, and $p_{out} = 0.01$, which has been shown in Figure 26.

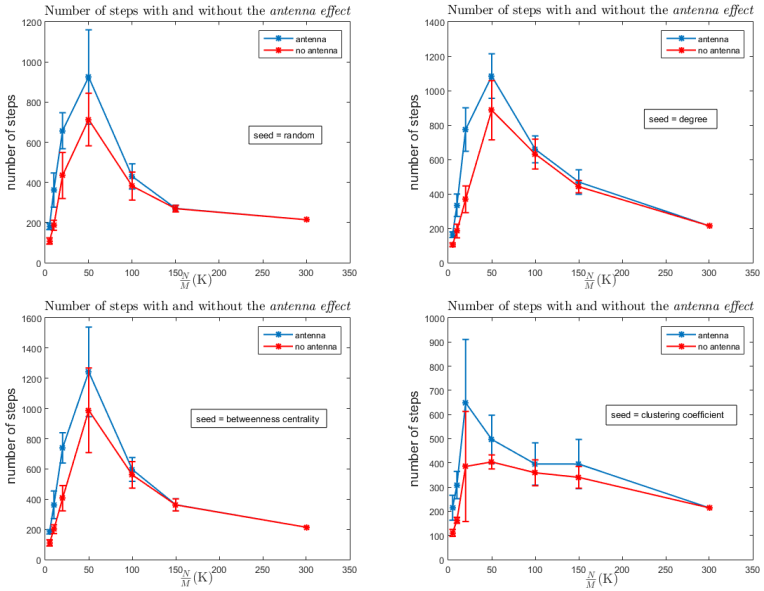

Figure 29: Planted partition graph with 300 nodes, $p_{in} = 0.2$ and $p_{out} = 0.01$. In blue: number of steps required by the original hierarchical method; in red: number of steps reassigning the nodes with degree 1 to other subgraphs.

We randomly select 2 *supernodes* (we require the nearest *supernode approach* to detect two subgraphs), which are the nodes numbered 20 and 49. They both belong to the first apparent cluster of the model, since nodes with index between 1 and 50 belong to the first apparent cluster, while nodes with index between 51 and 100 belong to the second apparent cluster. The subgraphs produced in the first phase of the hierarchical consensus method are reported in Figure 30. In this case, both subgraphs have a second–largest eigenvalue modulus of P equal to 0.97, while the second–largest eigenvalue modulus of the original graph is $\mu = 0.96$. Thus, in this pathological case, the first phase of the hierarchical consensus method is even slower than the non–hierarhical consensus method. This numerical example shows that the choice of the *supernodes* when a model with a cluster–exhibiting structure is considered, is extremely im-

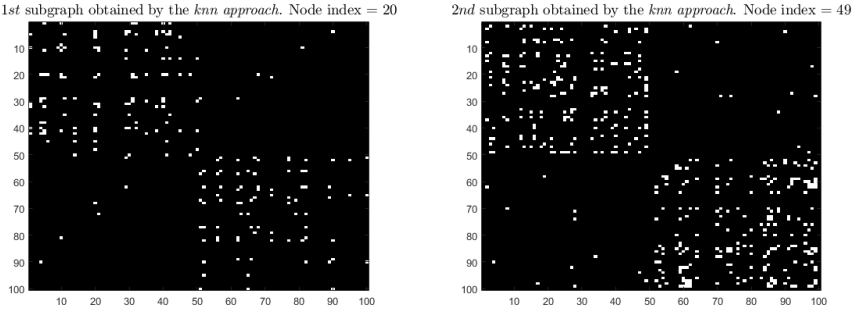

Figure 30: Subgraphs determined by the hierarchical method when the two *supernodes* belong to the same cluster.

portant. A possible way to overcome this problem could consist in making the *supernodes* change during the clustering process. If more clusters are present a more general technique that evaluates the dimension of the different clusters in order to correctly choose the *supernodes* is necessary. In addition, when more than 2 subgraphs are required to be detected, it is important to "balance" the *supernodes* in the different clusters. For instance, if we require the *nearest supernode approach* to determine 5 subgraphs, it is better to choose the *supernodes* such as 2 belong to a cluster and the other 3 to the other cluster. Again, a constraint on the choice of the *supernodes* evaluating their mutual distances is necessary.

4.7 Discussion and Conclusions

We studied the approximation of the global consensus state through a hierarchical consensus method, divided in two phases. The goal of the first phase is to extract, from the original agents' network, subgraphs with "good" spectral properties, which guarantee a fast convergence rate to the local consensus states of the subgraphs. In the second phase, an auxiliary graph, derived from such subgraphs, is considered, to find an approximation of the global consensus state. The proposed method has been motivated theoretically using spectral graph theory arguments

and also investigated numerically, comparing it with a non–hierarchical method on different kinds of graphs modeling real–world complex networks. The results of the proposed hierarchical consensus method are satisfactory in almost all the situations studied, showing usually a better performance with respect to the non–hierarchical method (i.e., a smaller number of iterations needed to guarantee the same accuracy in the approximation of the consensus state of the original network). We also investigated, both theoretically and numerically, a phenomen, called *antenna effect*, which could worsen, in some situations, the performance of the hierarchical consensus method itself. Then we suggested a solution to overcome the *antenna effect*, and demonstrated numerically its effectiveness. In fact, from both Figures 28 and 29, we can argue that the procedure suggested improves the results obtained by the original *nearest supernode approach*. Thus, the idea of reassigning the nodes with degree 1 to another subgraph where its degree is higher is promising and it can be applied to every type of graph. We also intend to better analyze, not only from a numerical point of view, the effect of connecting two different nodes with degree equal to one to a given graph. It would be interesting to study from a theoretical point of view, how the second–largest eigenvalue modulus changes when two nodes with degree equal to one are added to the same node and when they are connected to two different nodes.

In addition, we saw that when the *nearest supernode approach* is applied to compute the $1st$ phase of the hierarchical method, the best results are in general obtained when the clustering coefficient is chosen to select the *supernodes*. In fact, the nodes with highest clustering coefficient are expected to be a sort of "center" of a cluster, and the denser the subgraph the faster the convergence rate to the consensus state inside that subgrah.

As future studies we first intend to test the method with other types of graphs modeling real–world complex networks. At the same time we realized that depending on the structure of the graph, we have to decide which algorithm is better to apply during the $1st$ phase. Maybe it would be interesting to automatically determine the method to apply

during the $1st$ step. Additional studies and improvements are necessary for a better detection of the subgraphs, such as a completely automatic method able to choose the *supernodes* in the right clusters. Finally some improvements concerning the assignation of nodes in ON to nodes in SN can be evaluated.

4.8 Summary

In this chapter we concluded our study on the consensus problem. The common denominator between the previous chapter and this one is related to the study of the rate of convergence to the consensus state. In fact, in both the works we focused on the study of a method able to increase the convergence rate to the common opinion of a network of agents. In Chapter 3 we started from the algorithm developed in (48) adding to it different sparse variations with the aim of determining sparse solutions keeping as fast as possible the rate of convergence to the consensus state. In this chapter, instead, we directly focused on a method able to increase the convergence rate to the consensus state by decomposing the agents network in subgraphs with "good" spectral properties.

In the following chapters we continue our study on graphs–related techniques in a different context. Up to now we have studied graphs in the control field, now we focus on the machine learning field by evaluating and applying graph–based techniques in two different classification problems.

Part III

Graphs in semi–supervised learning and in modeling data for features extraction

5

Graphs in semi-supervised learning

5.1 Introduction

The properties and peculiarities of graphs are now exploited in a semi–supervised learning context. We apply a semi–supervised learning technique, called Laplacian Support Vector Machine (Laplacian SVM) (110), in the detection of flood–prone areas. Semi–supervised learning techniques are suitable in many situations and for different purposes. In fact, they are able to exploit the information coming from both the labeled and unlabeled data with the aim of obtaining a classifier that is better (in terms of performances) than a supervised classifier trained only on the labeled data. In many tasks, such as speech recognition, spam filtering, video surveillance and so on, there is a paucity of labeled data, because they require human annotators, special devices, or expensive and slow experiments; thus, techniques able to exploit the information coming from the unlabeled data are extremely useful (111). The detection of flood–prone areas is one of these examples, since the process of labeling

This chapter is partly based on:

- G. Gnecco, R. Morisi, G. Roth, M. Sanguineti, A. C. Taramasso, "Supervised and semi–supervised classifiers for the detection of flood–prone areas", Soft Computing, doi: 10.1007/s00500-015-1983-z

the data requires special devices, expensive and time–consuming experiments, or human annotators. The labeling process of the hazard level of a hydrological area, could be done, e.g., by a risk-analysis study performed by an expert, or by an analysis of historical series, which are both costly and time consuming processes. Thus, since in this context only a few labeled data are in general available, while a huge amount of unlabeled data is much more easier to be obtained, semi–supervised learning methods are suitable in this particular classification problem. For the reasons stated above we test the Laplacian SVM in the detection of flood–prone areas comparing its performances to the results obtained with a classical supervised learning algorithm such as Support Vector Machines (SVMs).

The chapter is organized as follows: Section 5.2 provides a brief introduction on semi–supervised learning focusing on the particular method adopted for our study, Section 5.3 presents the specific problem studied, while Section 5.4 presents the different experiments carried out distringuishing between the results achieved by an SVMs and by the semi–supervised technique Laplacian SVM; finally, Section 5.5, shows the results achieved and the relative discussion.

Note that, for the reader's convenient, we report the basic notations and definitions about SVMs in Appendix A, while the description of the Laplacian SVM is reported in Appendix B.

5.2 Semi–supervised learning: a brief overview

Let us consider a set of data x_i with $i = 1, \ldots, n$, belonging to a set $\mathcal{X} \subseteq \mathbb{R}^D$. Each sample x_i is represented by a D–dimensional *feature vector*, where each dimension is a *feature* of that particular sample. Without going into details, we recall that a generic learning task consists in predicting a value y_i, for $i = 1, \ldots, n$, called *label* for the specific *input* sample x_i. Thus, a generic learning algorithm aims at determining a function g

$$g : \mathcal{X} \to \mathcal{Y},$$

where the *output* domain $\mathcal{Y}$ is the *label* set, able to associate a *label*/*output* to every sample. In particular, $\mathcal{Y}$ can be either a discrete set (in the classi-

fication context) or a continuous one (in general, in a regression context). We refer, for instance, to (14; 112; 113) fur further details about the process usually followed in a machine learning context.

Our particular study is concerned with the classification of both labeled and unlabeled samples; we aim to determine a classification function g trained on both labeled and unlabeled samples able to predict the *output* of unseen *test* samples. Our dataset X is thus composed of l labeled samples $\{x_i, y_i\}_{i=1}^{l}$, provided with their *labels* y_i, and u unlabeled samples $\{x_j\}_{j=l+1}^{l+u}$, without the corresponding *label*; we fix $l + u = n$.

5.2.1 Manifold regularization

For the detection of flood–prone areas, we adopt a semi–supervised learning technique called manifold regularization, a technique developed in (110). A main assumption of manifold regularization is that the input data points are drawn from a probability distribution whose support resides on a Riemannian manifold embedded in the original feature space. A 2–dimensional manifold can be thought as a surface embedded in a higher dimensional Euclidean space (114). The surface of the Earth, for instance, is approximately a 2–dimensional manifold embedded in a 3–dimensional space. Similar remarks hold for larger dimensional manifolds. A Riemannian manifold is one on which one can define the "intrinsic distance" between any two points on the Riemannian manifold itself as their geodesic distance on the manifold, i.e., the length of the shortest path on the manifold between the two points.

Now, let us suppose that all the data (both the l labeled samples and the u unlabeled samples) are sampled from a probability distribution P on $X \times \mathbb{R}$. As previously mentioned, the assumptions on which the manifold regularization is based are two: the first one is that the marginal distribution P_X has support on a low dimensional manifold M embedded in the feature space X. The second is that if two points x_i and x_j are close with respect to the intrinsic distance on the Riemannian manifold they lie on, they are likely to have similar labels, i.e., the conditional probabilities $P(y|x_i)$ and $P(y|x_j)$ are similar. Here, with the term "intrinsic distance"

between two points we mean the length of the geodesic curve connecting them on the manifold. In practice, an approximation of the Riemannian manifold can be obtained by using both the labeled and unlabeled input data, building a weighted undirected graph $G = (V, E)$ with weighted adjacency matrix W (see e.g., Chapter 2) on the entire dataset considered. The graph here provides a discrete model of the manifold itself. In this particular context, assigning a weight to an edge means defining a measure of similarity between the associated vertices (input samples). Once a similarity measure has been chosen, the larger the similarity between the two input data points x_i and x_j, the stronger their connection in the graph. Thus, the higher the weight $w_{ij} = w_{ji}$ between x_i and x_j, the higher the probability that they belong to the same class. Determining a suitable similarity measure between every pair of input data points (hence, a suitable weight matrix W) is a challenging task, and several methods have been proposed in the literature to deal with such an issue. In fact, this measure is fundamental to build the graph that models the manifold where the data lie on. As described in Chapter 2 many similarity measures can be taken in consideration for the graph construction. In this particular context, we first consider a Gaussian similarity function $w_{ij} := e^{\frac{-||x_i - x_j||^2}{4t^2}}$ to define the weights of the edges. Then we define a *mutual k–nearest neighbor graph* (see, i.e., Section 2.2.2). As an example, Figure 31 shows the graph built on the specific data of the problem studied. In the figure, we consider only two features for each point in order to easily visualize the graph. In particular the graph represents an approximation of the Riemannian manifold where the data lie on. The graph has been built by using all the samples shown in the figure and by setting $k = 5$ in the definition of the *mutual k–nearest neighbor graph*.

5.3 Supervised and Semi-Supervised Classifiers for the Detection of Flood-Prone Areas

As mentioned in the introduction, we now aim at distinguish between flood–exposed and marginal–risk areas. In particular, we consider

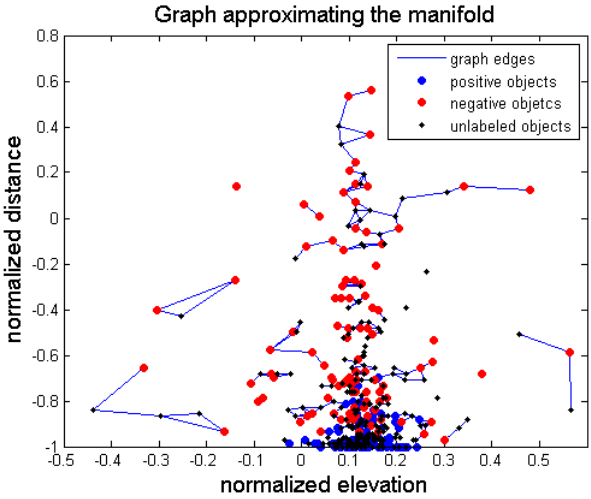

Figure 31: Example of a discrete approximation of the manifold the data lie one. The red points denote negative samples, the blue ones belong to the positive class, while the black points represent unlabeled samples.

Kernel–based binary classifiers (14) using six quantitative morphological features, derived from data stored in Digital Elevation Models (DEMs), comparing the recognition of the flood hazard obtained by both a supervised and a semi–supervised learning method. The use of such automatic classification techniques is valuable, e.g., in insurance applications, where one is interested in estimating the flood hazard of areas for which limited labeled information is available. In our particular situation, we intend to apply the machine-learning techniques to the basin of the Italian Tanaro River.

Among natural risks, floods are particularly relevant, as it is witnessed by the frequency of inundation events, together with all the associated negative consequences on society and local economies. The still limited availability of flood hazard information makes it very important to construct maps of flood-exposed areas, which should be one of the first steps in any analysis of the flood risk. In general, this kind of analysis begins with an investigation of hydrological data, and contin-

ues by modeling the evolution, in both space and time, of the flooding process itself (115; 116; 117). Unfortunately, techniques for the recognition of flood–exposed areas by experts down to very small scales (e.g., the scale of a single building) are very time–consuming and costly, as they require the acquisition of information which is not easily available for all the areas of interest, together with the intervention of the experts themselves. Hence, even nowadays, the mapping of flood–exposed areas is far from being complete. At the same time, in recent years, the availability of new technologies, such as radar and laser altimetry and GPS, has led to the development of Digital Elevation Models (DEMs), which have become standard tools of analysis for geomorphologists and hydrologists (118; 119; 120; 121; 122; 123). Since DEMs make automatically available several morphological and hydrological features (such as drainage areas, stream channels, and valley bottoms), they have replaced more time–consuming manual techniques. So, among other applications, nowadays DEMs are used for the identification of flood–exposed areas (see, e.g., (124; 125; 126)). The larger availability of measured data, compared to past acquisition techniques, has encouraged the application to flood risk analysis of machine learning techniques, too. This is done with the aim of using at the best the available information to construct flood hazard maps, or simply to suggest which areas should be subject to a more detailed investigation by experts (see, e.g., (127; 128)). The application of machine-learning techniques to the flood risk identification is particularly needed when a large portion of the data made available by DEMs is unlabeled. This is the case, e.g., of a high cost of labeling, as a detailed flood risk analysis by experts usually requires a specific study for each area of interest. Hence, we propose the application of the semi-supervised machine learning techniques Laplacian SVM to flood risk analysis, with the goal of making a better use of the large portion of unlabeled data which is often made available by DEMs. Likewise in (127; 128), we make use of a variety of morphological features for the detection of the local flood hazard, described later. The main difference with respect to the two above–mentioned papers consists in the additional exploitation of unsupervised information in the training of the

learning machine. At the same time, we compare the cases of supervised and semi–supervised training information. Likewise in (127; 128), the machine learning techniques are applied to the basin of the Italian Tanaro River. This has been selected as a case study due to the coexistence therein of different kinds of natural morphologies (from alluvial to mountain environments), together with the presence of civil and industrial settlements and their infrastructures. Indeed, the Tanaro basin includes environments that are representative of a much larger geographical region, which includes large portions of Italy and southern Europe, with the exception of arid areas and major rivers.

5.3.1 Flood–Prone Areas Dataset and Features

For the classification problem just introduced, we are provided with a dataset made of 187306 labeled data points belonging to the Tanaro basin area. Each data point is described by a set of 8 features ("feature vector"): its latitude (α) and longitude (β), its distance from the nearest stream (D), its elevation to the nearest stream (H), the local surface curvature (ΔH), the local contributing area (A), the local slope (S), and its absolute elevation (E). The data points are initially divided into two classes, one representing marginal–hazard areas, and the other one representing flood–prone areas. The first class is identified by the label 0. The data points belonging to the second class are further divided into three subclasses related to the hazard level of the area: the label $1\mathrm{l}$ stands for low-hazard level, $1\mathrm{m}$ for medium-hazard level, and $1\mathrm{h}$ for high-hazard level. Of the total 187306 labeled data points in the dataset, 55521 belong to flood–prone areas (without taking into account their hazard level), while the other 131785 come from marginal–hazard areas. The 55521 data points associated with flood–prone areas are further divided by hazard levels: 16659 are labeled as areas subject to a low flood hazard, 20099 are subject to a medium flood hazard, and the remaining 18763 ones are subject to a high flood hazard. In some of the experiments presented and discussed later, we will not distinguish among the 55521 flood-prone area data points according to their different hazard level, but we will group

them in the same class "flood–prone areas", giving them the label 1.

5.3.2 Classification tasks and experimental settings

We train several binary classifiers to learn to distinguish between data points belonging to two different classes of hydrologic areas. In particular, we consider the following binary classification problems: "marginal–hazard areas" versus "flood–prone areas", and "marginal–hazard areas" versus "high–hazard areas". This study is performed by using a subset of the dataset at our disposal for training/validation, and another subset for testing purposes. In more detail, the idea is to study, for different kinds of binary classifiers and different experimental settings, the classification performance of the trained classifiers, i.e., their capability to correctly identify the class associated with a data point given as input to the classifier. In doing this, we have to distinguish between data points used for training the classifier ("training samples"), and data points used as "test samples". A more detailed description of the kind of classifiers adopted in this study is reported in Section 5.4 and in Appendices A and B . Note that, while in (127) and (128) the investigations performed were limited to a restricted family of supervised classifiers and their focus was on the classification performance on only the training set, we intend now to exploit also unsupervised samples to train the classifiers.

In order to compare the classification performance of different classifiers, we perform several experiments to simulate and reproduce different realistic case studies. First, we consider two different binary classification problems, which differ for the choice of one of the two classes:

- binary classification of marginal–hazard (class 0) areas versus "high–hazard" (class 1h) areas ("0 vs 1h" classification problem);
- binary classification of marginal-hazard (class 0) areas versus "flood–prone" (class 1) areas ("0 vs 1" classification problem).

For each of such binary classification problems, we train and test the classifiers by using two different choices for the percentage of data points belonging to the high–hazard (respectively, flood–prone) class over the

total number of training/test data. More precisely, we consider the two following experimental settings:

- an unbalanced dataset is used, extracted from the original one, and characterized by a larger amount of high-hazard (respectively, flood–prone) data points than marginal–hazard data points: indicatively, 70% of high–hazard (respectively, flood–prone) training/test data points versus 30% of marginal–hazard training/test data points ("70% positive, 30% negative" case). This choice of the distribution of the positive and negative samples does not reflect the composition of the original dataset made of 187306 data points, but it is motivated by applications (such as insurance ones) where a classification error on a flood–prone area has more negative consequences than a classification error on a marginal–hazard area.

- a completely balanced dataset, which is again extracted from the original one: 50% of the training/test data is made of data points coming from high–hazard (respectively, flood–prone) areas, and the other 50% is made of marginal–hazard data points ("50% positive, 50% negative" case). In this context, the two classes of objects have the same importance.

In order to reduce the time required to train the classifiers and to keep the dimension of the training problem relatively small (hence, avoiding the use of more complex training procedures, optimized for huge datasets), for each kind of experiment we use only a relatively small subset of the entire dataset to train the classifiers. Implementation details are given later in Section 5.5.

5.4 The proposed learning approach

In order to simulate a real situation where few labeled data are available while a huge amount of unlabeled data is provided, we simulate possible realistic scenarios characterized by only a few number of labeled samples and a much larger number of unlabeled samples, with the aim

to determine a classification method that is able to deal efficiently with this kind of situation. In the specific, we consider n input data, x_i, with $i = 1, \ldots, n$, each one described by D features; thus, they are arranged in an input data matrix $X \in \mathbb{R}^{n \times D}$. In general, we will indicate with n the number of samples considered for each kind of experiment performed; thus, n can vary. However its value is always smaller or equal to 187306. D, instead, is smaller than or equal to 8 (details are provided later in Section 5.4.1). To each object x_i is associated its own label, i.e., the class it belongs to, which is denoted by y_i, $i = 1, , n$. Thus, we refer to each single data point as a pair $(x_i, y_i), i = 1, , n$. Since we always deal with binary classification problems, we can assume that either $y_i = 0$ or $y_i = 1$ holds. For instance, for the "0 vs 1" binary classification problem, the data points belonging to the three largest hazard levels are grouped into the same "positive" class 1, while the marginal–hazard data points compose the "negative" class 0. Hence, in this case, $y_i = 1$ for an input data point x_i corresponding to a flood–prone area, whereas $y_i = 0$ if x_i belongs to a marginal–hazard area. As described in Section 5.2, the labels y_i, with $i = 1, \ldots, n$ are collected into the n–dimensional output domain vector Y.

In order to simulate a realistic semi–supervised learning scenario, we further divide the dataset X in two subsets. The first one is represented by a matrix $X(L) \in \mathbb{R}^{l \times D}$ made of l labeled samples $\{(x_i, y_i), i = 1, \ldots, l\}$. To $X(L)$ we associate $Y(L)$, that is the vector containing the labels y_i, with $i = 1, \ldots, l$ corresponding to each x_i coming from the subset $X(L)$ of labeled samples. The second subset, instead, is represented by a matrix $X(U) \in \mathbb{R}^{u \times D}$ containing u unlabeled samples $\{x_j, j = l + 1, \ldots, l + u\}$. Clearly, the class of the u unlabeled objects is unknown to the classifier (particularly, in its training phase). The goal of semi–supervised techniques, such as the Laplacian SVM is to exploit potential class–related information coming from the unlabeled data to find a classifier that is better - in terms of the percentage of correctly classified data - than a classifier trained in a supervised way, using the labeled data alone (111).

We aim at building a classification model able to reach satisfactory

results for the hazard level classification in a "few labeled data points regime". At the same time, we are interested in obtaining a quite general classification model, one that can be used in as many situations as possible. Indeed, it is extremely useful to determine a classification model, trained on data belonging to a particular subregion of the river basin, that is able to perform well not only when tested on input data coming from the same subregion, but also from a different subregion of the same basin or even from another basin. In the following sections, we consider these two classification methods:

- a Laplacian SVM as a representative state of the art semi–supervised model (whose details can be found in Appendix B);
- a completely supervised SVM classifier, for comparison purposes. Appendix A reports a brief description of this learning method.

By simulating the Laplacian SVM semi–supervised scenario we shall investigate whether a satisfactory classification performance can be achieved by using only a few labeled samples, together with a much larger amount of unlabeled samples. This will be compared with the case where training is restricted to the labeled data above and a SVM is used, and to the situation in which also the originally-unlabeled data used to train the Laplacian SVM are provided to the SVM, together with their labels (of course, assuming that one can obtain such an information, at some additional cost). Since, in our context, the whole dataset is actually labeled, we do not provide to the classifier the labels of some training samples, in order to simulate the semi–supervised scenario. In this way, we deal effectively with them as unlabeled data, while we provide the remaining input data points to the classifier, together with their labels.

5.4.1 Experimental design

As mentioned in Section 5.3.1, each sample in the dataset is described by 8 features. However, when training the classifiers (both the supervised and the semi–supervised ones), we have decided not to provide the latitude and longitude features to them. Indeed, taking into account

such information would likely prevent the classifiers to learn a correct classification in regions geographically distant from the ones associated with the training data points. Moreover, in case of SVM/Laplacian SVM classifiers using certain kernel functions (see Appendix A), the latitude and longitude features are expected to be misleading for the classification of data points coming from sufficiently separated regions (e.g., this would happen in case of the linear kernel, see again Appendix A for its definition). However, as described later, in some cases we have still used the latitude and longitude features to separate geographically the training and test sets in a pre–processing phase, before training the classifiers. After having reduced in this way the number of features from 8 to 6, each feature is normalized in the interval $[-1, 1]$. This permits to give a–priori the same importance to each feature in the training phase. Then, different experiments are performed in order to test the performance of both the supervised and the semi–supervised classifier, dealing with case studies representing realistic situations. In particular, for each binary classification problem considered (i.e., either the "0 vs $1h$" or the "0 vs 1" problem), we run N different realizations of the experiments, as detailed in the following. First, the entire dataset is partitioned into two subsets S_1 and S_2, from which the training set and the test set are, respectively, extracted. This partitioning is obtained using the following three procedures.

- The first procedure consists in randomly assigning, with the same probability, each point of the entire dataset either to S_1 or S_2; clearly, in this way the two sets share no data.

- The second procedure exploits a threshold on the latitude feature to partition the input data. More precisely, the samples whose latitude is smaller than or equal to a given threshold are assigned to the set S_1, whereas the ones whose latitude is greater than the same threshold are assigned to the set S_2.

- The third procedure exploits a threshold on the longitude feature to partition the input data. The data points whose longitude is greater than or equal to a given threshold are assigned to the set S_1, while

those with longitude smaller than the same threshold are assigned to the set S_2.

Although no overlap between the sets S_1 and S_2 occurs for any of the three procedures, in the first case it may still happen that some data points in S_1 are very similar to data points in S_2, as they may belong to adjacent geographical areas. In the second case, however, this issue is likely limited to data points for which the latitude feature is near the threshold. A similar remark holds for the third case, referring to the longitude feature rather than the latitude one. Hence, the last two cases are intended to simulate a possible realistic situation where the sets S_1 and S_2 (hence, also the training and test sets) are composed of data belonging to different subregions. The procedure described below is followed to perform the different experiments.

Experiments pipeline

For each of the N realizations, a set TR is first randomly extracted from the set S_1, independently for each realization. Then, each type of classifier is trained taking a subset of TR as the training set (details are provided in Section 5.5), and tested on a different test set, which is also randomly extracted from the set S_2, independently for each realization. Thus, for each type of classifier, N training sets and N test sets are considered during the entire procedure. Finally, the classification errors computed at each realization, i.e., the "test errors", are averaged over the N realizations. When compared with the case of a fixed test set, the procedure of randomly generating N different test sets reduces the bias in the overall classification performance, producing results that are statistically more accurate. The SVM and Laplacian SVM classifiers considered in the following are characterized by a series of parameters to be tuned. Indeed, for both cases we have used a Gaussian kernel (see Appendix A), which contains an internal width parameter σ to be chosen. Similarly, for the Laplacian SVM one has to tune also the number k of nearest neighbors that are used to build the edge set E of the graph that is used inside the associated optimization problem (see Appendix B). Such pa-

rameters have to be tuned externally, as they assume fixed values in the optimization problems associated with SVM and Laplacian SVM training (see, respectively, Formula (A.1) and Formula (B.1)). In particular, for the SVM classifier with the Gaussian kernel, as it can be argued from Appendix A, the parameters are:

- the width σ of the Gaussian kernel (Formula (A.6));
- the regularization parameter γ_A (Formula (A.1)).

Similarly, as it is shown in Appendix B, the parameters of the Laplacian SVM classifier with the Gaussian kernel are:

- k: the number of nearest neighbors considered for the construction of the graph;
- t: the parameter used inside the definition of the Gaussian weights for the edges of the graph;
- the width σ of the Gaussian kernel (Formula (A.6));
- the regularization parameters γ_A and γ_I (Formula (B.1)).

For some of the parameters, we then perform a cross–validation procedure with K folds (112; 113) in order to find the values that are most suitable for the particular context and situation investigated. In particular, after having determined the best set of parameters, i.e., the one corresponding to the smallest average validation error, the model is trained again on the entire training set, using such optimal parameters. Finally, the obtained model is evaluated on the test set, which has not been previously used in the overall training/validation process. In particular, during the cross–validation procedure, the training samples are equally distributed in the K folds in order to make each fold as representative as possible of the entire training set. More precisely, in order to have the same percentage of labeled and unlabeled data in each group, at first we divide all the u unlabeled objects in K folds, assuming that u is a multiple of K. Similarly, both the positive and negative labeled samples (l_P objects for the "positive" class, l_N objects for the "negative" class) are

partitioned into K different groups, assuming that both l_P and l_N are multiples of K. Thus, each fold is composed of $\frac{l_P}{K}$ positive labeled samples, $\frac{l_N}{K}$ negative labeled samples, and $\frac{u}{K}$ unlabeled samples. In order to highlight the effect of the unsupervised samples when moving from the SVM to the Laplacian SVM classifier, the parameters that are in common to both the classifiers have been assigned to the same values for both models. More precisely, first, the value of the width σ of the Gaussian kernel is fixed. Then, the cross–validation procedure detailed above is performed in order to find the best value for the regularization parameter γ_A for the SVM. Subsequently, the parameters σ and γ_A of the Laplacian SVM are fixed to the same values chosen for the SVM (i.e., its a–priori fixed valued for σ, and the one γ_A chosen by the first cross–validation performed for the SVM). Moreover, the value of the Gaussian weight parameter t of the Laplacian SVM is fixed to the same value chosen for the parameter σ, as the two parameters have similar meanings. Then, for the Laplacian SVM, the cross–validation is performed to find the best values for the remaining parameters k (the number of nearest neighbors used to build the associated graph) and γ_I (the additional regularization parameter), which are involved in the "semi–supervised component" of the optimization problem associated with the Laplacian SVM, see Formula (B.1). The motivation for the whole procedure is that both parameters σ and γ_A appear in the "supervised component" of such an optimization problem (as Formula (B.1), and Formula (A.6) show), which is in common with the optimization problem associated with the SVM (see Formulas (A.1) and (A.6)). As the procedure assigns the same values to such common parameters, any difference in classification performance between the two classifiers is likely to be ascribed mainly to the absence/presence of unsupervised training samples.

5.5 Results and Discussion

5.5.1 Experimental results

We implement the method described in Section 5.4 in MATLAB. The code is mainly based on the library lapsvm from (129), available at `http://sourceforge.net/projects/lapsvmp/`. We report the results obtained by generating, for each type of experiment, $N = 10$ realizations. As mentioned in Section 5.4.1, the test set is randomly and independently generated at each realization, and is made of $T = 4000$ samples. In particular, in the "70% positive, 30% negative" case it is made of 2800 samples from the positive class and 1200 from the negative one. Similarly, still in the "70% positive, 30% negative" case, the training set, randomly extracted during each realization, is composed of 140 labeled samples for the positive class, 60 labeled samples for the negative class, and $u = 3800$ unlabeled samples, which are generated from labeled samples simply by removing their labels before presenting them to the classifier. In addition, 2660 of these unlabeled samples come from the positive class, and 1140 from the negative (but this additional information is not provided to the classifier). In the "50% positive, 50% negative" case, instead, the test set is made of 2000 samples for each class, and the training set is composed of 100 labeled samples for each class, and $u = 3800$ unlabeled samples, which are generated as described before. In addition, 1900 of these unlabeled samples come from the positive class, and 1900 from the negative one (and again, this additional information is not provided to the classifier). Concerning the completely supervised SVM classifier, in the following tables we report two kinds of classification results. The first row shows the results achieved by the SVM trained only with $l = 200$ labeled data (the same labeled samples used also to train the Laplacian SVM), while the third row reports the results obtained by training the SVM on all the 4000 data composing the training set (i.e., reinserting the labels in its unlabeled samples). Finally, the second row refers to the semi–supervised Laplacian SVM, trained with $l = 200$ labeled data and $u = 3800$ unlabeled data. As regards the choices of the parameters in the classifiers, the width σ of the Gaussian kernel is set

to 0.5, and also the parameter t of the Gaussian edge weight is set to 0.5. Concerning the cross–validation procedure, we fix a number of folds $K = 5$. Then, a first cross–validation is performed on the regularization parameter γ_A, restricting the values considered for such a parameter in the three–elements set $\{10^{-3}, 10^{-2}, 10^{-1}\}$. As already mentioned, we decide to fix the parameters σ and γ_A to the same values for both classifiers in order not to have substantial differences in the supervised part of the optimization problems associated, respectively, with the completely supervised SVM classifier, and the semi–supervised Laplacian SVM classifier. Then, a second cross–validation procedure is performed for the additional parameters of the Laplacian SVM model. In the specific experiments, the following possible choices for such parameters have been examined during cross–validation: the number k of neighbors needed for the graph construction in the Laplacian SVM is chosen inside the three–elements set $\{5, 7, 10\}$, whereas the value of the second regularization parameter γ_I is chosen inside the three–elements set $\{10^{-1}, 1, 10\}$. Likewise the set used for γ_A, such sets are chosen of small cardinalities, to limit the computational time needed to perform the cross–validation procedure. Tables 1–6 show the results achieved by both the supervised and semi–supervised classifiers ("pos" stands for "positive", while "neg" for "negative"). We report the results obtained for the different case studies described in the sections above. The tables show the mean accuracies obtained averaging the test accuracies over the $N = 10$ different realizations, and their empirical standard deviations. Such accuracies are defined, respectively, as

- the percentage of data points in the test set correctly classified by the trained classifier (accuracy);
- the true positive rate *TP*, which is the ratio between the number of "positive" samples in the test set that are correctly classifed by the trained classifier as "positive" samples, and the total number of "positive" data in the test set;
- the true negative rate, defined as *TN* = 1 - *FP*, where FP stands for false positive rate, i.e., the ratio between the number of "negative"

Table 1: "0 vs 1_h" binary classification problem, when the sets S_1 and S_2 are obtained by a random partitioning of the dataset.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$93\% \pm 0.5\%$	$95\% \pm 0.3\%$	$89\% \pm 1\%$
		LapSVM	$92\% \pm 0.4\%$	$98\% \pm 0.5\%$	$78\% \pm 0.2\%$
		SVM_{4000}	$96\% \pm 0.2\%$	$98\% \pm 0.4\%$	$90\% \pm 2\%$
50% **pos**	50% **neg**	SVM_{200}	$92\% \pm 1\%$	$93\% \pm 4\%$	$91\% \pm 2\%$
		LapSVM	$91\% \pm 0.5\%$	$95\% \pm 1\%$	$87\% \pm 1\%$
		SVM_{4000}	$95\% \pm 0.4\%$	$96\% \pm 0.4\%$	$93\% \pm 0.6\%$

Table 2: "0 vs 1_h" binary classification problem, when the sets S_1 and S_2 are defined, respectively, as the subset of objects with latitude greater than or equal to 2750 (expressed in pixel units), and the subset of objects with latitude smaller than the same threshold.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$89\% \pm 2\%$	$97\% \pm 5\%$	$75\% \pm 10\%$
		LapSVM	$91\% \pm 1\%$	$99\% \pm 0.5\%$	$73\% \pm 5\%$
		SVM_{4000}	$95\% \pm 0.8\%$	$97\% \pm 0.9\%$	$91\% \pm 2\%$
50% **pos**	50% **neg**	SVM_{200}	$89\% \pm 2\%$	$90\% \pm 5\%$	$89\% \pm 5\%$
		LapSVM	$91\% \pm 1\%$	$98\% \pm 1\%$	$83\% \pm 4\%$
		SVM_{4000}	$94\% \pm 0.9\%$	$95\% \pm 3\%$	$92\% \pm 1\%$

samples in the test set that are erroneously classified by the trained classifier as "positive" samples, and the total number of "negative" data in the test set.

As an illustrative example, Figure 32 shows the ground truth. Figure 33, instead, reports on the left the classifications produced by an SVM trained on $l = 200$ labeled samples, while on the right the results produced by a Laplacian SVM trained on $l = 200$ labeled samples and $u = 3800$ unlabeled samples when the scenario considered is the one reported in Table 4, with 50% positive training samples and 50% training

Table 3: "0 vs 1_h" binary classification problem, when the sets S_1 and S_2 are defined, respectively, as the subset of objects with longitude smaller than or equal to 2400 (expressed in pixel units), and the subset of objects with longitude greater than the same threshold.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$89\% \pm 3\%$	$90\% \pm 0.6\%$	$87\% \pm 3\%$
		LapSVM	$91\% \pm 0.8\%$	$99\% \pm 0.7\%$	$75\% \pm 3\%$
		SVM_{4000}	$94\% \pm 0.6\%$	$97\% \pm 1\%$	$89\% \pm 1\%$
50% **pos**	50% **neg**	SVM_{200}	$88\% \pm 2\%$	$88\% \pm 5\%$	$90\% \pm 3\%$
		LapSVM	$90\% \pm 0.7\%$	$96\% \pm 2\%$	$84\% \pm 3\%$
		SVM_{4000}	$93\% \pm 0.3\%$	$96\% \pm 0.7\%$	$91\% \pm 0.6\%$

Table 4: "0 vs 1" binary classification problem, when the sets S_1 and S_2 are obtained by a random partitioning of the dataset.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$89\% \pm 1\%$	$90\% \pm 2\%$	$86\% \pm 2\%$
		LapSVM	$90\% \pm 0.7\%$	$93\% \pm 1\%$	$80\% \pm 2\%$
		SVM_{4000}	$93\% \pm 0.4\%$	$98\% \pm 0.3\%$	$77\% \pm 2\%$
50% **pos**	50% **neg**	SVM_{200}	$88\% \pm 0.3\%$	$90\% \pm 2\%$	$86\% \pm 1\%$
		LapSVM	$87\% \pm 0.3\%$	$92\% \pm 2\%$	$83\% \pm 2\%$
		SVM_{4000}	$90\% \pm 0.5\%$	$93\% \pm 0.5\%$	$86\% \pm 0.7\%$

Table 5: "0 vs 1" binary classification problem, when the sets S_1 and S_2 are defined, respectively, as the subset of objects with latitude greater than or equal to 2750 (expressed in pixel units), and the subset of objects with latitude smaller than the same threshold.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$86\% \pm 5\%$	$93\% \pm 8\%$	$68\% \pm 8\%$
		LapSVM	$90\% \pm 1\%$	$99\% \pm 1\%$	$65\% \pm 8\%$
		SVM_{4000}	$90\% \pm 2\%$	$98\% \pm 3\%$	$65\% \pm 2\%$
50% **pos**	50% **neg**	SVM_{200}	$80\% \pm 2\%$	$93\% \pm 5\%$	$67\% \pm 7\%$
		LapSVM	$83\% \pm 3\%$	$98\% \pm 1\%$	$68\% \pm 8\%$
		SVM_{4000}	$83\% \pm 2\%$	$96\% \pm 4\%$	$70\% \pm 2\%$

Table 6: "0 vs 1" binary classification problem, when the sets S_1 and S_2 are defined, respectively, as the subset of objects with longitude smaller than or equal to 2400 (expressed in pixel units), and the subset of objects with longitude greater than the same threshold.

			Accuracy	TP	TN
70% **pos**	30% **neg**	SVM_{200}	$87\% \pm 2\%$	$88\% \pm 4\%$	$83\% \pm 5\%$
		LapSVM	$88\% \pm 1\%$	$91\% \pm 2\%$	$80\% \pm 2\%$
		SVM_{4000}	$89\% \pm 0.7\%$	$95\% \pm 0.7\%$	$69\% \pm 2\%$
50% **pos**	50% **neg**	SVM_{200}	$84\% \pm 2\%$	$89\% \pm 5\%$	$79\% \pm 5\%$
		LapSVM	$85\% \pm 0.5\%$	$91\% \pm 1\%$	$79\% \pm 2\%$
		SVM_{4000}	$83\% \pm 0.5\%$	$92\% \pm 0.7\%$	$74\% \pm 0.8\%$

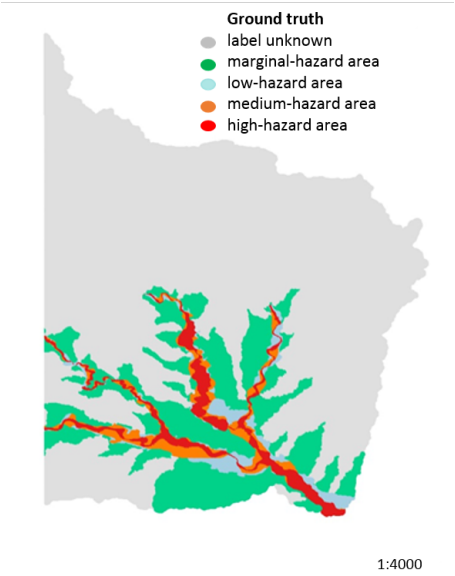

Figure 32: Ground truth.

negative samples, and the sets S_1 and S_2 are obtained by a random partitioning of the entire data set. Concerning the supervised classifier, the results are obtained by choosing an SVM with the best parameters (in terms of the validation error) among the N different classifiers trained during the N realizations; while, the semi–supervised classifier chosen is the one of the N different semi–supervised classifiers trained during the N realizations with the "best" set of parameters, in terms of the validation error.

Note that Figure 33 reports the classification results obtained by the two best classifiers on a larger dataset, which contains also samples whose labels are completely unknown (not only to each learning machine, during its training), simply because their labels are not available in such an extended dataset. So, they differ from the 3800 "unlabeled" samples used by the Laplacian SVM, which, as already reported, are originally "labeled" samples, whose label has only been "hidden" to the learning

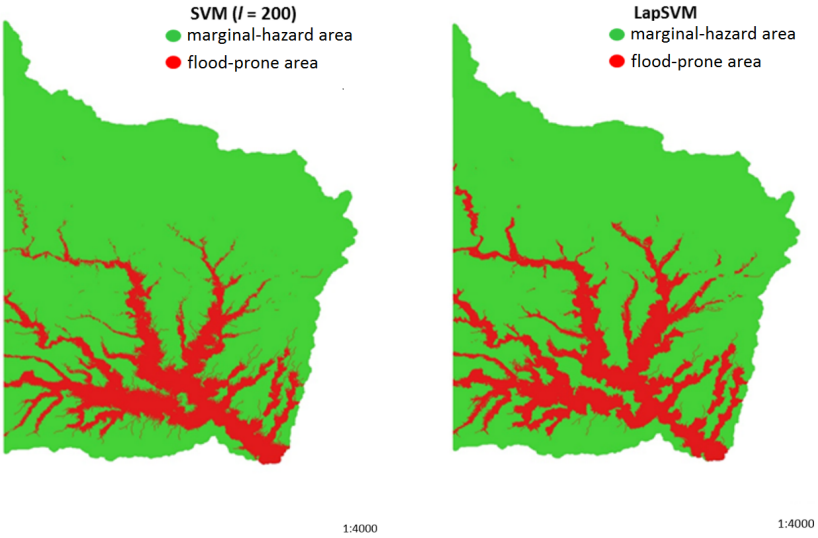

Figure 33: Output of the SVM classifier ($l = 200$) on the left. Output of the Laplacian SVM on the right.

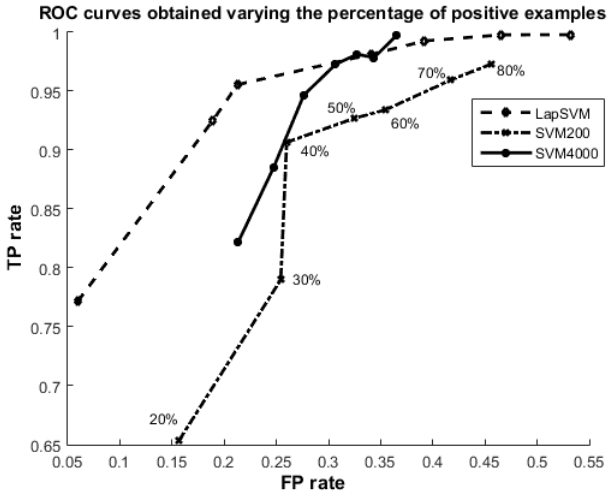

Figure 34: ROC curves of the three classifiers computed by changing the percentage of the positive samples over the total number of samples.

machine during its training.

In addition, to better investigate the behavior of the TP and FP rates on the test set, we evaluate their variations with respect to the composition of the training set, focusing on the case in which the three classifiers are applied to the "0 vs 1" binary classification problem, and the sets S_1 and S_2 are defined by means of a threshold on the latitude (i.e., the same scenario of the results reported in Table 5). In particular, we vary the percentage of positive and negative samples in both the training and test sets, which are both randomly extracted from S_1 and S_2. More precisely, we consider 7 different situations where the percentage of positive samples over the total is $20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%$, respectively. For each type of situation, $N = 10$ different realizations are considered and both the TP and FP rates are averaged over the N realizations. We display the obtained results by means of the Receiver Operating Characteristic (ROC) curve (130) shown in Figure 34. In such figure, the percentage of positive samples over the total is indicated only for the curve representing the results of the SVM trained with $l = 200$

labeled samples. Clearly, the same labeling of the percentage holds for the other two curves.

5.5.2 Discussion and Conclusions

In this work, the semi–supervised technique, Laplacian Support Vector Machine (Laplacian SVM), has been studied and applied for the classification of marginal–hazard and high-hazard/flood prone areas. The emphasis is on understanding the potentiality of this semi–supervised method when only few labeled samples are provided, together with a much larger number of unsupervised samples. In order to highlight the capabilities of the Laplacian SVM classifier, we performed different experiments comparing the performances obtained by this technique with the ones achieved by the fully supervised SVM classifier, using the same number of labeled data. For comparison purposes, we considered also the case of a SVM classifier trained with a much larger number of (costly) labeled samples. To understand the generalization capability of the models in various situations, we took into account different cases for the generations of the training and test sets. Beside choosing the training and test set from the same sub–region, in other experiments we separated the training and test sets by using longitude and latitude information, in order to simulate a possible situation where the classifier is trained with data belonging to a specific sub–region, and subsequently tested on a different sub–region.

Tables 2–6 show a Laplacian SVM accuracy equal or even larger than the SVM accuracy (except for the case shown in Table 4, "50% positive, 50% negative"), when the latter is trained with the same number of labeled samples. The improvements are larger in the unbalanced case, where one 70% of the test set is made of positive samples, while the remaining 30% is made of negative samples. Moreover, we can see that the Laplacian SVM accuracy always improves when moving from the totally balanced situation ("50% positive, 50% negative") to the unbalanced ("70% positive, 30% negative") one (except for the case shown in Table 2, where the accuracy remains the same). In addition, the True Pos-

itive (TP) rates reported in the different tables highlight that the Laplacian SVM performs better than the SVM trained with the same number of labeled samples in detecting the actual high–hazard/flood prone areas. This is particularly important for applications such as insurance ones. In this context, indeed, the wrong classification of a positive sample as a negative one is much less desirable than the wrong classification of a negative sample as a positive one. Thus, if one has to choose between a higher TP rate with a slightly higher FP rate, and a lower TP rate with a slightly lower FP rate, the first case is preferred. A smaller percentage of positive samples are in fact misclassified in the first case. However, the results reported in the third row of the tables show that a better classification performance is usually obtained for the SVM, when this is trained using a number of labeled samples equal to the number of (labeled and unlabeled) data employed by the Laplacian SVM itself. This is expected, as such a SVM models a case in which the classifier has at its disposal a larger amount of (costly) information for its training.

In particular, from Tables 5 and 6 one realizes that, for the "0 vs 1" binary classification problem, the Laplacian SVM achieves the same or even a better performance than the SVM trained with $l = 4000$ labeled samples, when the two sets S_1 and S_2 are obtained by using a threshold either on the latitude or on the longitude and both are trained by using 50% positive samples and 50% negative samples. Hence, the semi–supervised classifier shows a larger generalization capability when the training and test sets are extracted from geographically separated regions. In addition, from Figure 34 we can infer that the Laplacian SVM performs well also when unbalanced data sets are considered, and the error in the detection of the high–hazard/flood–prone areas is more penalized than the error in the detection of marginal–hazard areas. In general, it could be argued that the differences in the performances of the Laplacian SVM and SVM, reside on a larger generalization capability of the semi–supervised method. This is due to the fact that this model takes into account more samples to model the manifold where the data lie on, hence acquiring useful information related to the space and the relationships between the different samples. Note that, this information do not

require any additional cost, since the data are unlabeled. Thus, the Laplacian SVM is, in general, able to perform better than the SVM, especially when training and test are extracted from geographically separated regions and when unbalanced sets are considered. In the data analysis, we used a variety of morphological features for the detection of the local flood hazard. In addition, the results show that for each type of scenario studied (training and test sets randomly extracted from the dataset or obtained by a threshold on the latitude/longitude), the overall accuracy of every type of classifier is higher in the "0 vs $1h$" binary problems than the "0 vs 1" ones. This behavior is expected since in the "0 vs 1" binary problems more similar samples belonging to different classes are present. We expect that the classification between samples from low–hazard level areas and samples from flood–prone areas is more difficult than the classification between flood–prone areas and high–hazard level areas. The samples from the latter scenario are supposed to be more different, thus easier to be classified, than the ones belonging to the first classification scenario.

It is worth noting that, as a future extension, we intend to improve the methodology adopted in the different experiments and to run additional tests to better validate the semi–supervised technique applied to the specific problem. More specifically, the idea is to compute the performance of both the semi–supervised and supervised methods by changing the distribution of the positive and negative samples in the training set, keeping fixed the distribution of the samples in the test set. In addition, we intend to examine an alternative construction of the ROC curve displayed in Figure 34 considering the same classifier and evaluating its performances obtained for various distributions of the samples, and correspondingly optimized thresholds. Finally, to obtain more accurate and complete results it would be interesting to include the parameter t of the Gaussian weight present in the Laplacian SVM as a parameter to be optimized during the cross–validation procedure.

In addition, one can take into account also externalities modeled through features related to the local human intervention against floods. We focused on binary classifiers, as our goal consisted in comparing different

classifiers based on the absence/presence of labels associated with some samples (dealt with as supervised by some classifiers, and as unsupervised by other ones), without taking into account the number of possible values for the labels. Another future development consists in extending the comparisons to the multi–class case, either training directly multi–class classifiers, or combining several binary classifiers, likewise in (127).

5.6 Summary

In this chapter we have studied the utilization of graphs in a semi–supervised learning context in order to evaluate their capabilities in a specific classification problem. In particular we focused on the classification of flood–prone areas that represents a typical classification problem characterized by a large number of data, the majority of them unlabeled. In fact, the labeling process is often expensive, time–consuming and it requires special devices. For this problem, a semi–supervised learning method able to exploit the information coming from the unlabeled data is thus extremely important. We saw that the Laplacian SVM outperforms the SVM in almost all the situations studied. Hence, the promising results obtained highlighted the potentiality of graphs in determining a discrete approximation of the manifold on which the data reside. The manifold is built on both the labeled and the unlabeled samples and it allows to obtain additional information on the data and to achieve better performances than the ones obtained by a supervised technique computed by using only the labeled data.

In the next chapter we will continue our evaluation and study of graphs in a machine learning context. In particular we will deal with graphs with the aim of modeling the connections among the samples in a dataset. The goal is to extract additional features of each single sample in the dataset that can be useful to improve the classification accuracy of a classifier trained on the original dataset without considering graph–based features. We will apply graphs in a medical context, with the aim of automatically distinguishing patients with different forms of Parkinsonisms.

6

Graph-based features extraction in a supervised learning context

6.1 Introduction

In this chapter we continue the study related to the application of graphs in a machine learning context. While in the previous chapter graphs are exploited to obtain a classifier trained on both labeled and unlabeled samples with better performances of a classical supervised classifier trained on the labeled data only, the idea now is to exploit graph in the features computation process. More precisely, we will deal with binary and multi–class classification problems in a supervised learning context. The idea is to extract additional information, i.e., features, from

This chapter is partly based on:

- R. Morisi, G. Gnecco, N. Lanconelli, S. Zanigni, D. Manners, C. Testa, S. Evangelisti, L. L. Gramegna, C. Bianchini, P. Cortelli, C. Tonon, R. Lodi, "Binary and multi–class Parkinsonian disorders classification using Support Vector Machines", Lecture Notes in Computer Science, 2015,
- R. Morisi, G. Gnecco, N. Lanconelli, S. Zanigni, D. Manners, C. Testa, S. Evangelisti, L. L. Gramegna, C. Bianchini, P. Cortelli, C. Tonon, R. Lodi, "Binary and multi–class classification of parkinsonian disorders with support vector machines based on quantitative brain MR and graph–based features", submitted to Movement Disorders.

each sample starting from a graph built on the dataset, with the aim of improving the classification performances obtained when the samples described by only their original features (without the ones obtained from the graph) are provided to the same supervised classification model.
The problem we are dealing with concerns the classification of Degenerative Parkinsonisms. We have already studied pattern recognition problems in the medical field. In particular, in (131; 132) we focused on the detection of myocardial scars in patients who suffered of cardiac problems. The goal was to provide to the clinicians a technique able to automatically determine either the presence or the absence of scars in the myocardium of a patient. In that particular problem, we applied pattern recognition techniques on segmented Magnetic Resonance Images. Then, in (133) we also carried out another study that again applied patter recognition techniques in the medical field. It concerned the application of an unsupervised learning method, Independent Component Analysis (ICA), for the detection of ischemic territories in the heart. ICA was applied to timeseries extracted from the heartbeat, with the aim of automatically distinguishing timeseries coming from healthy territories and ischemic territories in the heart.
Now we proceed with the study of machine learning methods applied to the medical field. We consider again the classical SVM as the supervised learning technique, applying it to a dataset made of patients with parkinsonian disorders with the aim of automatically distinguishing the different disorders. Beside the learning method, we intend to model the dataset by means of a graph in order to extract information and additional knowledge about the relationships among the patients either with the same disorder or with different disorders.
Details related to this specific issue are provided in Section 6.2, then in Section 6.3 we describe the dataset we are dealing with and how the different features are computed; in Section 6.4 we introduce the application of graphs in this particular context and the classification method adopted, while the results obtained and the conclusions are provided in Section 6.5.1 and 6.5.2, respectively.

6.2 Degenerative Parkinsonisms - overview

Degenerative Parkinsonisms, such as Idiopathic Parkinson's Disease (PD), Progressive Supranuclear Palsy (PSP) and Multiple System Atrophy (MSA), with the cerebellar (MSA–C) and parkinsonian (MSA–P) variants, are chronic progressive diseases characterized primarily by movement impairment accompanied by various degrees and combinations of autonomic, cognitive and behavioral alterations. Although these disorders are characterized by different clinical features, response to pharmacological treatment and prognosis, the in vivo differential diagnosis is often challenging because of clinical overlapping and definite diagnosis can be reached only post–mortem. In order to improve the accuracy of clinical diagnostic criteria, various promising biomarkers have been identified (134). In particular, advanced Magnetic Resonance (MR) quantitative markers of brain microstructure, metabolism, regional atrophy respectively obtained from Diffusion Tensor Imaging (DTI), proton spectroscopy (^{1}H-MRS) and morphometric-volumetric analysis, demonstrated high accuracy in differentiating parkinsonian syndromes (135). (136) showed that DTI can be useful in classifying subjects with Parkinson's disease, atypical parkinsonism and essential tremor, while (137) and (138) investigated the potentiality of MR Spectroscopy in the study of Parkinson's Disease. A multimodal approach combining different quantitative MR markers may also improve diagnostic accuracy and may be useful to discriminate parkinsonian syndromes at an individual level. In addition, a pattern recognition approach able to provide a first quick and automatic diagnosis, helping the experts for its successive refinement, should be extremely useful. Recently, Support Vector Machines (SVMs) have been applied in this context to discriminate parkinsonian syndromes at an individual level, using a combination of multiple features, i.e., various MR markers. In particular, in (139) SVMs have been used in distinguishing patients with PD from those with PSP, in (140), the automatic classification of PD patients from normal volunteers has been studied, while (141) has applied SVMs for the automatic discrimination of PD from atypical forms of Parkinsonism. Another study where

SVMs have been applied in a related context is described in (142), where this method is used to discriminate between essential tremor with rest tremor and tremor–dominant Parkinson Disease. On the other hand, (143) presents a multi–class classification problem applying a statistical technique for the discrimination of different forms of Parkinsonism (PD, PSP, MSA–C, and MSA–P). Multi–class classification problems are studied also in (144) together with binary problems, performing the analysis by means of SVMs, for both the binary and the multi–class classification problems.

Now, our idea is to start from the results and type of study performed in (144), and to investigate the possibility and the potentiality of computing additional features to provide to the machine learning method in order to better discriminate the different disorders. We intend to exploit a graph-based technique on the data at our disposal with the aim of collecting additional information related to the connections among the different patients. Stronger connections and similarities of features are expected among patients with the same disorder rather than patients with different disorders. Thus, a graph built on the markers of each single patient is expected to provide such information. In addition, as performed in (144), we intend to apply and study a feature selection algorithm in order to better understand the most discriminative subset of features, comparing the results with the experts' opinions. The feature selection algorithm is also expected to provide useful information about the importance of the graph–based features computed and added to the original dataset.

6.3 Patients and Methods

In this section, we provide a description of the dataset at our disposal. In particular, we report in Section 6.3.1 some information about the different patients; instead Section 6.3.2 describes the process of acquisition and computation of the different markers, starting from the MR images.

Table 7: Demographic and clinical features of the study sample.

	Mean ± standard deviation			
variable	**PD**	**PSP**	**MSA–C**	**MSA–P**
Sex: No. of M/F	$32/16$	$12/10$	$5/4$	$6/1$
Age at the evaluation (years)	64.7 ± 10.2	73.5 ± 6.5	59.1 ± 7.0	61.6 ± 10.6
Disease duration (years)	3.8 ± 3.2	3.5 ± 1.9	6.8 ± 3.9	4.9 ± 2.9

6.3.1 Dataset preprocessing

We include in the study 9 MSA–C, 7 MSA–P, 22 PSP and 48 PD consecutive patients who underwent brain MR at the Functional MR Unit of the Policlinico S. Orsola - Malpighi, in Bologna, Italy, as part of the diagnostic work–up. Clinical diagnosis were performed according to current criteria (145; 146). Their demographic data are summarized in Table 7.

We are provided of $l = 86$ (i.e., the total number of patients considered) labeled pairs $(x_i, y_i), i = 1, \ldots, l$, where $x_i \in \mathbb{R}^D$ ($D = 152$ MR markers/features are provided for each patient) and $y_i \in \{1, 2, 3, 4\}$ (4 different types of disorders are considered). Thus, the patients are arranged in a dataset $X \in \mathbb{R}^{l \times D}$, while the labels of each single patient are arranged in a l–dimensional feature vector Y. In particular, $l_1 = 9$ samples belong to class 1, $l_2 = 7$ belong to class 2, $l_3 = 22$ to class 3, while $l_4 = 48$ samples are from class 4.

Note that, before being provided to the supervised learning technique for the disorders classification, the dataset X has been preprocessed re–scaling each feature inside the interval $[-1, 1]$ in order to give them the same importance a–priori.

6.3.2 MR Imaging

All participants underwent the same brain MR protocol with a 1.5 T GE scanner: axial T2–weighted FLAIR, coronal T2–weighted FSE, T1–weighted volumetric FSPGR, axial Diffusion Tensor Imaging (DTI) with 25 directions and a single–voxel cerebellar proton-MR spectroscopy (^{1}H-MRS) using PRESS sequence. A manual morphometric analysis was ex-

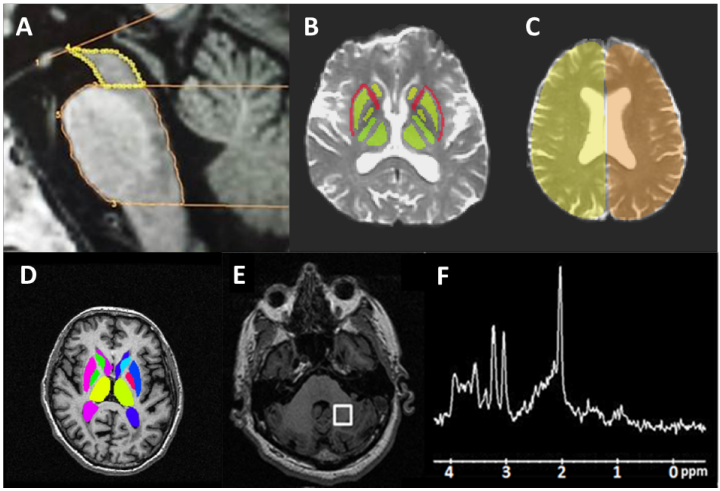

Figure 35: Examples of imaging and spectroscopic quantitative parameters used as features for SVM analysis: manual morphometry (A); DTI FA and MD quantification by ROIs (B) and histograms (C) analysis; semi–automatic segmentation of deep brain structures (D); cerebellar volume of interest localization (E) and corresponding ^{1}H–MR metabolites spectrum (F).

ecuted in order to measure midbrain area, pons area, the diameters of the middle and superior cerebellar peduncles (MCP and SCP), their ratios and the MR parkinsonism index (MRPI) (147) (7 features). The mean DTI parameters (Fractional Anisotropy, FA, and Mean Diffusivity, MD) were calculated by using a Regions of Interest (ROIs) method (84 features) and by using an histogram–analysis method (148) (34 features). A semi–automated volumetric and mean DTI parameters analysis based on FSL tools `http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/` was performed for deep brain structures, lateral ventricles, cortical lobes, and cerebellum (21 features). We quantified in (^{1}H-MRS) spectra N–Acetyl Aspartate (NAA), Choline (Cho), Creatine (Cr) and myo–inositol (mI) content expressed as ratios: NAA/Cr, Cho/Cr, mI/Cr, NAA/mI (4 features). The fitting–program LCModel v 6.3 was used to perform the last analysis. An example of processed MR images is shown in Figure 35.

6.4 Pattern Recognition Analysis

In this particular context, the potentialities of SVMs in both binary and multi–class classification problems have been previously studied in (144). We first perform the same type of study by considering a larger dataset, and we then apply a graph–based technique to extract additional features. In the binary classification case, we consider all the possible combinations of pairs of disorders (named "one disorder vs another") with the goal of distinguishing the two classes; at the same time we perform binary classification problems between a specific disorder and all the other ones grouped in a single class, we refer to this case as "one disorder vs all". Moreover, the potentiality of SVMs are evaluated in a four–class classification scenario, where each disorder is treated as a single class, and in a three–class classification scenario, considering MSA–C and MSA–P as a single class, since they are subtypes of the same disorder.

The SVMs performances are evaluated and compared when they are applied to different sets of features. More precisely, as a reference, we first consider the original dataset provided by the clinicians where each single patient is described by $D = 152$ MR markers evaluating the results achieved by the learning method; then we add the additional graph–based features to the original dataset with the aim of improving the performances of the SVMs.

In the following, we first provide the description of the graph–based model built on the dataset in order to extract additional features for each single patient, then the supervised classification method used in this particular context is presented and, finally, we provide a brief description of the feature selection technique adopted to determine the most meaningful subset of MR markers and graph–based features for this particular problem.

6.4.1 Modeling the dataset with graphs

Our idea is to extract additional features from the dataset X in order to add to the original $D = 152$ features more information that can be

useful to improve the classification performance of the SVMs. In particular we build an undirected unweighted graph $G = (V, E)$ (see Chapter 2) on the MR markers. For our purposes, a graph built on the dataset made up of the MR markers of the patients can provide useful information about the similarities between patients with the same Parkinsonism and dissimilarities between patients with different parkinsonian disorders. Each node $v_i \in V$ with $i = 1, \ldots, l$ represents a patient. We expect, indeed, that two patients v_i and v_j with the same disorder are much more similar, in terms of the values of the MR markers collected, rather than two patients with different disorders. Hence, in the graph G, they are expected to be connected by an edge e_{ij}, while edges between patients with different disorders are expected to be more rare. Thus, a graph whose measure of similarity is built on the MR markers is expected to highlight connections among the different patients. We build a *k–nearest neighbor graph* (see, i.e., Chapter 2), by first computing the euclidean distance between all possible pairs of nodes (each patient is seen as a point in the euclidean space, whose coordinates are given by the values of the markers). Then, the number k of desired connections, i.e., edges, for each node is fixed and two nodes v_i and v_j, are connected by an edge if either v_i is among the k–nearest neighbors of v_j or vice–versa. The final adjacency matrix associated to the graph is a binary one. Once the graph is built, 6 additional features are computed for each node; these features are subsequently added to the feature set made by the MR markers of each patient. The additional features extracted from each node are mainly measures of centrality that give information of the structural importance of a node, i.e., a patient, in the graph. In particular, they are: the degree, the local average degree (which gives information about the degrees of the neighboring nodes of a given node), the closeness and betweenness centrality (which measure the influence a node has on the other nodes), the eccentricity of a node and the clustering coefficient. It is reasonable to assume that these additional features added to the original dataset should be useful in the overall classification process since we can expect that nodes, i.e., patients, belonging to the same class have similar values of the same feature, while nodes, i.e., patients, in different classes have

different values of the same feature. More precisely, for each binary classification problem we expect that, on average, the degree of the nodes belonging to the class with more patients is higher than the one of the nodes belonging to the less numerous class. We recall that the graph is a *k–nearest neighbor graph*, thus, in general, there are more than k edges for each node. In addition, considering, for instance, the closeness centrality, it can be expected that nodes belonging to the most numerous class have a closeness centrality higher than nodes belonging to the less numerous class. In fact, since this feature, for each node v_i, is obtained by dividing 1 for the sum of the lengths of the shortest paths between v_i and all the other nodes, we can easily expect that this sum is smaller for the nodes belonging to the most populous class than the other one. Similar arguments can be provided for the other graph–based features computed. In general, these types of features give information about the interactions each node is involved in; fur further details, please refer to Chapter 2.

6.4.2 Classification model

For our classification problems, we choose an SVM with linear kernel. In fact, by running some preliminary comparison tests with other kernels (e.g., polynomial, radial basis function), it turned out that the linear SVM performed better than the others. Indeed, it is not unusual that in high–dimensional datasets with a small number of samples, radial basis function and polynomial kernels lead to overfitting, while the linear kernel provides the best result (149).

The model adopted in this study is a classical binary classification problem, with l samples x_i, $i = 1, \ldots, l$, and the corresponding labels $y_i \in \{-1, 1\}$. Later, the generalization of this model used to deal with the multi–class classification problems is described. Starting from the general formulation of a binary SVM described in Formula A.3, the opti-

mization problem for the binary linear SVM is the following:

$$\begin{aligned} &\text{minimize}_{w,b,\xi} \quad \left(\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i\right) \\ &\text{subject to} \qquad y_i f(x_i) \geq 1-\xi_i, \text{ for } i=1,\ldots,l\,, \\ &\qquad\qquad\quad \xi_i \geq 0, \text{ for } i=1,\ldots,l\,, \end{aligned} \tag{6.1}$$

where $C > 0$ is a penalty term and $f(x_i) = w^T x_i + b$ is an affine function of x_i. Beside the model described by Formula (6.1) which assigns a sample x_i to one of the two classes according to the sign of $f(x_i)$, a probabilistic output can be also used (150). More precisely, given a sample x_i and the corresponding value of $f(x_i)$, the a–posteriori probability of x_i to belong to one of the two classes $y_i \in \{-1, 1\}$ can be estimated by a sigmoid function in the following way:

$$P(y_i = j|x_i) = \frac{1}{1+\exp(Af(\mathrm{x_i})+B)}, \tag{6.2}$$

where j is either equal to -1 or 1, and A and B are suitable parameters. Then, x_i is automatically assigned to the class y_i with the highest posterior probability between the two. The parameters A and B in formula (6.2) are estimated by solving a regularized maximum likelihood problem, following the procedure described in (151). The probabilistic output reveals to be especially suitable in the multi–class classification scenario. In that case, to automatically classify the samples according to the different classes, we use a combination of R two–classes 1–vs all classifiers, where R represents the number of classes presented, i.e., $R = 4$ in the four–class classification study, while $R = 3$ when MSA–C and MSA–P are considered as a single class. More specifically, R binary 1–vs all linear SVMs with probabilistic output $P_r(x)$ ($r = 1, \ldots, R$) are trained, i.e., for each class the training is realized by labeling the samples belonging to the class chosen as samples from the "positive" class, while all the others are grouped together in the "negative" class. Once the R classifiers are trained, a new input x is tested by presenting it to all the R classifiers, then it is assigned to the class with the highest a-posteriori probability,

i.e., the one that maximizes $P_r(x)$:

$$\text{maximize}_r \; p(y = r|x).$$

For each type of binary classification problem considered, we adopt the following pipeline:

- a binary problem is chosen, either a "one disorder vs another" case or "one disorder vs all" situation;
- depending on the dimension of the dataset considered, a series of values k for the k–nearest neighbors procedure adopted to build the graph are chosen. For instance, if one is interested in the classification problem "MSA–C vs MSA–P", the values for k are smaller than the values considered in a binary classification problem that takes in consideration the entire dataset, such as "one disorder vs all". In fact, in this case, the dimension of the original dataset is larger than the one in the previous situation, thus a larger number of connections between the different nodes (the samples) can be fixed;
- a set of values for the regularization parameter C is fixed;
- the classification process is performed:
  - if graph–based features are not considered, we run a leave–one–out cross–validation procedure (LOOCV) to determine the penalty term C value (see Formula (6.1)) that leads to the best accuracy, computed as the average of the samples correctly classified at each run of the LOOCV procedure;
  - if graph–based features are added to the original dataset, the LOOCV procedure is performed on both the values k of the number of nearest neighbors considered to build the graph and on the values for the penalty term C. The accuracy is again computed by averaging over the samples correctly classified at each run of the LOOCV procedure determining the k and C values that lead to the best result;

Note that the process described above is performed with both the entire set of features and only a subset extracted from the original dataset by a feature selection algorithm described in the following section. In addition, we are not considering an independent evaluation test set since our dataset is relatively small; the accuracy reported in the results in Section 6.5 is then the one associated to the best C value obtained by averaging the results on the samples that are left out at every step of the LOOCV procedure. At each iteration of the LOOCV a value 1 is considered if the classifier correctly classifies the sample left out, 0 otherwise. The predictions are then averaged over the total number of samples. In addition, beside the best results of the overall classification process (the one corresponding to the best C value), in order to have a more accurate estimate of the final results, we report the average and standard deviation of the classification process obtained with the entire set of the values of the regularization parameter C considered for each binary classification problem.

Concerning the multi–class classification scenarios, the pipeline followed is the same as the one described above. One of the two multi–class classification problems has to be chosen, either the one with 4 classes or the one with 3 classes. Then, again a set of values for the parameter k and C is fixed and the LOOCV procedure is performed determining the best accuracy for each problem considered.

Note that, for both the binary and the multi–class classification problems, the C value and the number of nearest–neighbors k fixed to build the graph, may depend on the classification problem itself. In other term, different classification problems may have different optimal values for C and k.

6.4.3 Feature selection

As a pre–processing step, a feature selection method is applied to the dataset in all the problems treated; we aim at understanding if some features are more meaningful and useful than others for diagnosis purposes. In addition, it is interesting to see if the most meaningful MR

markers automatically selected, agree with the ones considered by the experts opinions as the most useful ones. We apply a ranking criterion to the D features in the dataset X which orders the features according to their capability of assigning each sample to the correct class. In particular, for each binary classification problem, an independent evaluation criterion is used, i.e., the features are considered individually in determining their capability to discriminate between the two classes, when each feature is used alone. We choose the relative entropy criterion (152) as the method to assess the ability of every feature in separating the two labeled classes. First, its value is computed for each of the D features. Then, the features are ranked decreasingly with respect to their relative entropies, following the procedure described in (112). Depending on the subsequent classification problem one is interested in, the $\tilde{D} < D$ features corresponding to the $\tilde{D}$ best values of the ranking can be subsequently selected, to form a reduced dataset $\tilde{X} \in \mathbb{R}^{l \times \tilde{D}}$ (this procedure is repeated for several choices of $\tilde{D}$). Then, in each binary classification scenario, we evaluate the outcome of the feature selection method by means of Receiver Operating Characteristic (ROC) curves (153). By comparing the ROC curve obtained with only $\tilde{D}$ features corresponding to the $\tilde{D}$ best values of the ranking, and the one for the entire set of D features, an estimate of the goodness of the feature selection method can be done. We propose to build the ROC curve by changing the value of the threshold on the probabilistic output of the SVM. More precisely, when a classifier, already trained, is tested on a new sample x, the output obtained is the probability p of x to belong to the "positive" class and $1 - p$ to belong to the "negative" class, or in other words, in the default case, the threshold is set equal to $\frac{1}{2}$. Then, by means of a threshold on p, the sample x is assigned to one of the two classes. The classifier automatically assigns x to the "positive" class if $p \geq 1-p$, to the negative class otherwise. Changing the threshold on p, different classification rules, thus different classification results can be obtained and a ROC curve can be determined.

All the described methods are implemented using MATLAB and, specifically for the classification part, we use the codes described and implemented in (150).

6.5 Results and Discussion

6.5.1 Binary and multi–class classification results

We first investigate two binary classification scenarios, the first one considers all the combinations between pairs of different disorders, i.e., PSP vs. PD, MSA–C vs. MSA–P and so on, while the second one consists in classifying each disorder versus the other three, grouped in a single class. Subsequently, we consider two multi–class classification problems: a 4–class problem (i.e., the number of classes is equal to the number of disorders), and a 3–class one (the classes MSA–C and MSA–P are here grouped into one single class, as being variants of the same disorder). Also for the binary classification problems, we perform some tests by considering MSA–C and MSA–P as a single class, named as MSA.

These situations are first evaluated and studied by applying the SVMs on the dataset made of the MR markers alone (i.e., the additional graph–based features are not used in this first analysis). The results achieved are considered as a sort of reference in order to subsequently compare such results with the performances obtained when additional graph–based features are added to the dataset. In particular, for each type of classification problem, both the binary and the multi–class one, a graph is first built on the specific dataset involved, then 6 more features are computed for each patient and added to the original set of features. An example of the adjacency matrix obtained on the entire dataset is shown in Figure 36; in particular, for this example, a number of nearest neighbors $k = 40$ is fixed. The light–blue squares presented in the figure highlight the 4 different classes. For both the binary and the multi–class situations we compare the performances of the linear SVM obtained with the entire set of features and with only a small subset extracted by the ranking procedure described in Section 6.4.3. A LOOCV procedure is performed in both the situations (i.e., a dataset with either D or $\tilde{D}$ features), to determine the best C and k values for each type of problem. In particular, 100 values equally spaced in the interval $[10^{-2}, 1]$ are considered for the regularization parameter C. In addition, in the reduced–dimension scenario, we compute the accuracy of the SVM by considering a reduced dataset

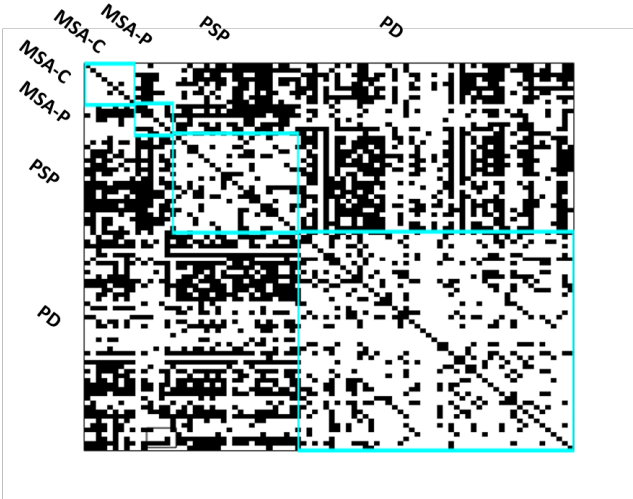

Figure 36: Adjacency matrix when the entire dataset is considered, fixing the number of nearest neighbors $k = 40$.

made of the first $\tilde{D} = 10$, $\tilde{D} = 20$, $\tilde{D} = 30$ and $\tilde{D} = 40$ ranked features.

First of all we report some results concerning the study of the feature selection technique. Figure 37 shows a comparison in terms of ROC curves, between the outcomes obtained with the entire set of D features and a reduced set made of the first $\tilde{D} = 20$ ranked features in the "PD vs MSA" binary problem. In the default case of the value $\frac{1}{2}$ for the threshold, the accuracy achieved when graph–based features are not added to the dataset (plot on the left) is equal to 91% when the entire set of $D = 152$ is considered, while it is equal to 92% when the first $\tilde{D} = 20$ ranked features are taken in consideration. Regarding the result achieved when the graph–based features are considered, the accuracy is equal to 91% when the entire set of $D = 158$ is used, while it is equal to 94% when the first $\tilde{D} = 20$ ranked features are considered. Note that, by an inspection of the ROC curves, it is possible to set a value of the threshold different from the one considered by default by the SVM (i.e., $\frac{1}{2}$), that leads to a higher accuracy. Nevertheless, since we are interested in an automatic classification procedure, we prefer to maintain the threshold equal to $\frac{1}{2}$ for the computation of the final accuracy. In this way, the method is as general

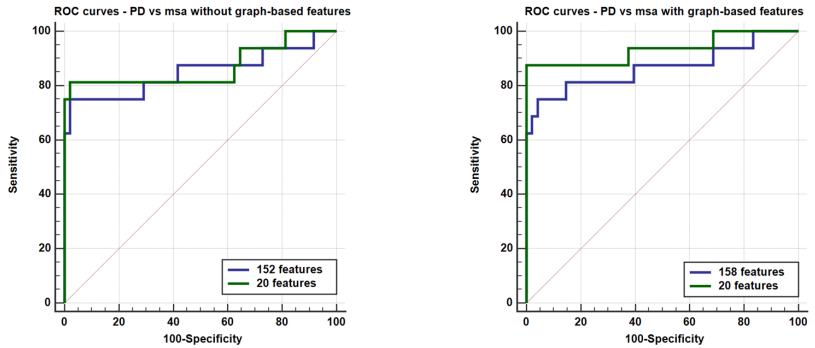

Figure 37: Comparison between the ROC curves obtained when the entire set of features is considered (blue curve) and when only a subset with the first 20 ranked features is used (green curve). On the left the results without using graph–based features, on the right the curves obtained when graph–based features are added to the dataset.

and automatic as possible.

Figure 38 shows the frequency of the first 40 features extracted by the ranking procedure in the 13 binary problems "one disorder versus all" and "one disorder versus another" (whose results are reported in Tables 9 and 12, respectively) when graph–based features are not taken in consideration. In dark are highlighted the most used features, i.e., those appearing among the first 40 ranked features in at least 10 binary problems over the 13 total. Specifically, they are: Magnetic Resonance Parkinsonism Index (MRPI), MD Middle Cerebellar Peduncle (MCP) right (R), MD cerebellar White Matter (WM), MD pre–frontal WM left(L), MD pre–frontal WM R, MD Posterior Fossa (PF) (25° percentile), MD Posterior Fossa (PF) (50° percentile), MD cerebellar hemisphere R (25° percentile), MD cerebellar hemisphere R (50° percentile), MD cerebellar hemisphere L (50° percentile), MD cerebellar hemispheres R+L and cerebellum volume. Note that, 43 features over a total of 152 are never listed in the first 40 features, suggesting their negligible importance in the specific classification problem.

From the results obtained, we notice that the best accuracy (the one

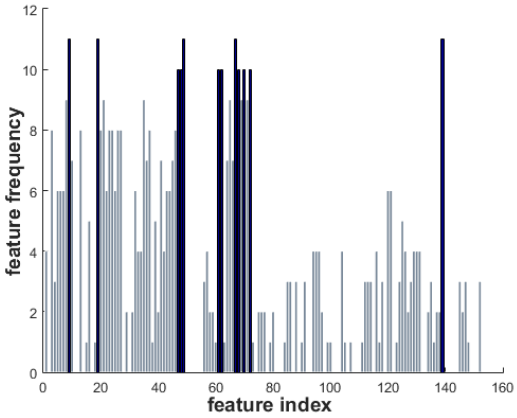

Figure 38: Frequency of the first 40 ranked features in the 13 binary classification problems "one disorder vs another" and "one disorder vs all".

reported in Tables 9 and 12 for the binary classification problems, and in Tables 18 and 19 for the multi–class classification scenarios) in most of the cases is achieved when only a small subset of features is considered. Considering the binary classification scenarios, when graph–based features are not added to the dataset, only two problems need the entire set of features, i.e., the "MSA–P vs PSP" and the "MSA vs all" problems, while 5 binary classification problems achieve the best accuracy only with the first 10 ranked features, 2 problems need 20 features, 2 the first 30 and the remaining 2 binary problems achieve the best accuracy when the first 40 ranked features are considered. When graph–based features are taken in consideration, instead, only the classification problem "MSA–P vs PSP" needs the entire set of features to reach the best accuracy. In 4 binary classification problems the best accuracy is achieved by considering only the first 10 features, whereas 20 features are used in 2 problems, and 30 in other 6 situations.

A selection of the results achieved by the SMVs by both considering and not considering graph–based features varying the number of features selected by the ranking procedure is shown in Table 8.

Table 8: Accuracy in the binary classification problem "PD vs another" varying the dimension of the set of features considered.

	PD vs PSP		PD vs MSA–C		PD vs MSA–P		PD vs MSA	
# features	No graph	Graph	No graph	Graph	No graph	Graph	No graph	Graph
10	96%	99%	96%	96%	91%	91%	88%	94%
20	96%	99%	98%	98%	89%	93%	92%	94%
30	94%	96%	98%	100%	89%	93%	92%	95%
40	94%	94%	98%	100%	91%	93%	92%	94%
all	94%	99%	98%	98%	87%	87%	91%	91%

Secondly, we report the accuracy, sensitivity (TP rate) and specificity (TN rate) (154) of each type of binary classification problem considered. In particular, as previously mentioned, we first study each type of scenario by considering the entire set of features and then we perform the same study considering a dataset made of only the first $\tilde{D} = 10, 20, 30$ or 40 first ranked features. We decide not to consider datasets with a number of features more than 40 because we consider at most 40 features to be a good compromise between keeping the dimension of the problem relatively small and obtaining a good accuracy. The results reported in Tables 9 and 12 are the best among all these possibilities, according to the LOOCV procedure described above. In addition, to better investigate the results achieved with the other values of the parameter C, we also report the mean and standard deviation of the accuracies of each type of binary classification problem related to every value of C. In particular, Tables 10 and 13 report the results when graph–based features are not considered. While Tables 11 and 14 show the mean and standard deviation of the accuracies overall the entire set of values of the parameter C when graph–based features are taken in consideration. Note that, in this case, the results reported correspond to the optimal value of k.

Table 9 reports the accuracy, sensitivity and specificity when SVMs are used to discriminate between a disorder "positive class" and all the other disorders grouped in a single class "negative class". Column on the left reports the results when the classifier is applied on the MR markers alone, while the right column reports the results achieved by adding graph-based features to the dataset.

Table 12 reports the results of the binary classification problems "one

Table 9: Accuracy, sensitivity and specificity of SVMs in binary classification problems "one disorder vs all". Left column: results obtained without graph-based features; right column: results obtained by using graphs to extract more features to provide to the classifiers. The stars specify when a statistically significant improvement is obtained when using graph–based features.

	No graph			Graph		
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
PD vs all	87%	90%	84%	94%	94%	95%
PSP vs all	98%	91%	100%	98%	91%	100%
MSA–C vs all	95%	67%	99%	98%	89%	99%
MSA–P vs all*	92%	0%	100%	98%	71%	100%
MSA vs all*	94%	56%	100%	98%	88%	100%

Table 10: Mean and standard deviation over the entire set of values of the regularization parameter C of the accuracies of the binary problems "one disorder vs all". Graph–based features are not considered.

	PD vs all	**PSP vs all**	**MSA–C vs all**	**MSA–P vs all**	**MSA vs all**
Accuracy	$86\% \pm 0.7\%$	$97\% \pm 0.4\%$	$95\% \pm 0.7\%$	$92\% \pm 0\%$	$91\% \pm 1\%$

Table 11: Mean and standard deviation over the entire set of values of the regularization parameter C of the accuracies of the binary problems "one disorder vs all". Graph–based features are added to the original dataset.

	PD vs all	**PSP vs all**	**MSA–C vs all**	**MSA–P vs all**	**MSA vs all**
Accuracy	$92\% \pm 3\%$	$96\% \pm 0.8\%$	$96\% \pm 1\%$	$96\% \pm 1\%$	$96\% \pm 1\%$

Table 12: Accuracy, sensitivity and specificity of SVMs in binary classification problems "one disorder vs another". Left column: results obtained without graph-based features; right column: results obtained by using graphs to extract more features to provide to the classifiers. The stars specify when a statistically significant improvement is obtained when using graph–based features.

	No graph			Graph		
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
PD vs PSP	96%	99%	91%	99%	98%	100%
PD vs MSA–C	98%	100%	89%	100%	100%	100%
PD vs MSA–P*	91%	98%	43%	93%	98%	57%
PD vs MSA*	92%	100%	69%	95%	100%	81%
PSP vs MSA–C	100%	100%	100%	100%	100%	100%
PSP vs MSA–P	97%	100%	86%	97%	100%	86%
PSP vs MSA	97%	100%	94%	97%	100%	94%
MSA–C vs MSA–P	94%	100%	86%	94%	100%	86%

Table 13: Mean and standard deviation over the entire set of values of the regularization parameter C of the accuracies of the binary problems "one disorder vs another". Graph–based features are not considered.

	PD vs PSP	**PD vs MSA–C**	**PD vs MSA–P**	**PD vs MSA**	**PSP vs MSA–C**
Accuracy	$95\% \pm 0.8\%$	$95\% \pm 1\%$	$89\% \pm 0.7\%$	$90\% \pm 2\%$	$97\% \pm 1\%$
		PSP vs MSA–P	**PSP vs MSA**	**MSA–C vs MSA–p**	
Accuracy		$96\% \pm 0.5\%$	$95\% \pm 1\%$	$73\% \pm 7\%$	

disorder vs another". Again the SVMs are applied both to the original dataset (column on the left) and on a dataset increased of 6 features extracted from the graph built on the dataset of each type of problem considered (column on the right).

As previously mentioned, the LOOCV procedure is performed in all the situations studied (both binary and multi–class classification problems) in order to determine the values for the regularization parameter C and the number of neighbors k used to build the graph. We report in Table 15 the optimal number k used to build the graph and subsequently to extract additional features in the binary problems "one disorder vs another", while Table 16 shows the optimal values for k used to achieve the best accuracy in the binary problems "one disorder vs all". Finally, the

Table 14: Mean and standard deviation over the entire set of values of the regularization parameter C of the accuracies of the binary problems "one disorder vs another". Graph–based features are added to the original dataset.

	PD vs PSP	PD vs MSA–C	PD vs MSA–P	PD vs MSA	PSP vs MSA–C
Accuracy	$97\% \pm 1\%$	$98\% \pm 0.4\%$	$90\% \pm 2\%$	$95\% \pm 1\%$	$97\% \pm 1\%$
		PSP vs MSA–P	**PSP vs MSA**	**MSA–C vs MSA–P**	
Accuracy		$96\% \pm 1\%$	$95\% \pm 1\%$	$6\% \pm 7\%$	

Table 15: Optimal number of k nearest neighbors and features, with the corresponding best accuracy, for the binary problems "one disorder vs another".

	Accuracy	**k**	**# features**
PD vs PSP	99%	40	10
PD vs MSA–C	100%	30	30
PD vs MSA–P	94%	15	30
PD vs MS	95%	30	30
PSP vs MSA–C	100%	5 / 10	10
PSP vs MSA–P	97%	10	158
PSP vs MSA	97%	10 / 15 / 20	10 / 20
MSA–C vs MSA–P	94%	30	15

optimal value for k in the multi–class classification problems are reported in Table 17. We also provide the number of features (right column) considered to achieve the best accuracy, that is, again, reported in the first column of the Tables. Note that in some situations, such as in the "PSP vs MSA" problem or in the "PSP vs all" binary classification problem, more than one value for k are indicated. In fact, it happens that in many cases, different values for k lead to the best result; we decide to report only some examples together with the optimal number of features.

Concerning the multi–class classification scenario, the results are reported by means of a confusion matrix in Tables 18 and 19. In particular, Table 18 refers to the case where graph–based features are not added to the dataset. The table on the left reports the results in the 4–class classification problem, when all the parkinsonian disorders are considered

Table 16: Optimal number of k nearest neighbors and features, with the corresponding best accuracy, for the binary problems "one disorder vs all".

	Accuracy	k	# features
PD vs all	94%	30	30
PSP vs all	98%	20 / 35	40 / 30
MSA–C vs all	98%	30 / 35	30 / 20
MSA–P vs all	98%	40	10
MSA vs all	98%	30 / 35	30 / 20

Table 17: Optimal number of k nearest neighbors and features, with the corresponding best accuracy, for the two multi–class classification problems.

	Accuracy	k	# features
4–classes	88%	60	40
3–classes	94%	35	20

as a single class, while the table on the right shows the results of the 3–class classification problem, when MSA–C and MSA–P are grouped in one class, the MSA class. On the other hand, Table 19 reports the results achieved when the 6 graph–based features are added. In particular, the accuracy of the classifier in the 4–class classification problem is, in both the situations, with and without the graph–based features, equal to 88% and it is achieved when the first 40 ranked features are considered. Concerning the 3–class classification problem, without adding features from the graph, an accuracy of 91% is obtained with a dataset of 40 features, while the best accuracy achieved when graph–based features are added is obtained with the first 20 ranked feaures and it is equal to 94%.

6.5.2 Discussion and Conclusions

The proposed pattern recognition analysis has been revealed to be extremely suitable for the type of classification problem considered. First of all our results allow to conclude that SVMs applied to the original dataset made of the MR markers provided by the clinicians are able to achieve satisfactory results in all the binary classification problems con-

Table 18: Confusion matrices: on the left the 4–class classification problem, on the right the 3–class classification problem. Graphs are not used to extract additional features.

		Predicted			
		MSA–C	MSA–P	PSP	PD
Actual	MSA–C	8	0	0	1
	MSA–P	1	1	0	5
	PSP	0	0	20	2
	PD	0	0	1	47

		Predicted		
		MSA	PSP	PD
Actual	MSA	11	0	5
	PSP	0	20	2
	PD	0	1	47

Table 19: Confusion matrices: on the left 4–class classification problem, on the right the 3–class classification problem. Graph–based features are added.

		Predicted			
		MSA–C	MSA–P	PSP	PD
Actual	MSA–C	8	0	0	1
	MSA–P	1	1	0	5
	PSP	0	0	20	2
	PD	0	0	1	47

		Predicted		
		MSA	PSP	PD
Actual	MSA	14	0	2
	PSP	0	20	2
	PD	0	1	47

sidered. Satisfactory results are even obtained by the combination of R two–classes 1–vs all classifiers for the multi–class classification problems. In particular, we observed that our results are in line (in terms of results and accuracies achieved) with the previous works that consider binary classification problems of parkinsonian disorders, such as (139), (140) and (141). In fact, (140) applied an SVM approach to brain MR structural and resting state fMRI data obtained from 19 PD patients and 27 normal volunteers finding an accuracy of 86.96% in correctly discriminating the two conditions. In (141), the authors applied the same method to DTI data analyzed with a Tract Based Spatial Statistics (TBSS) pipeline from 17 PD patients and 23 with other forms of parkinsonism. They found that an SVM including 100 features discriminated PD patients at an individual level with high accuracy (97.5%). Finally, in (139) a single–class SVM classification was applied to structural T1–w and DTI images obtained from 21 PSP and 57 PD patients, finding an accuracy of 100% by using the WM atrophy at each voxel as predictor.

At the same time, concerning the multi–class classification scenario,

we achieved results in line with (143) where an accuracy equal to 84.5% in the 4–class classification problem was obtained, while, concerning the 3–class classification problem, an accuracy of 91.7% was achieved. In addition, comparing the results obtained without adding graph–based features to the ones achieved in (144), where we considered a subset of the dataset used for this work, the results improve in 7 situations, the accuracy remains the same in 3 problems, while it decreases in 3 classification scenarios. When graph–based features are added, additional improvements with respect to the results achieved in (144) are obtained, since the accuracy remain the same in 3 problems, while it increases in the remaining 10 binary classification scenarios. Regarding the present work, when graph–based features are extracted from the original dataset, in 8 binary problems out of 13, the accuracy, thus the sensitivity and specificity, improve with respect to the results obtained when graph–based features are not considered. In particular in 4 cases the improvement is statistically significant (we evaluated the results by computing the Area Under the Curve of the ROC built on the results achieved when only the MR markers are provided to the classifier and when graph–based features are added, following the procedure developed in (155)). The statistically significant improvements are indicated with a * in Tables 9 and 12. Concerning the multi–class classification problems, adding graph–based features to the MR markers led to improvements in the 3–class classification scenario. In addition, compared to the multi–class classification problems studied in (144), the results obtained in the present work improve in both the scenarios with and without graph–based features.

To our knowledge, the application of a graph in extracting additional features from a dataset represents a novel approach in this particular medical problem. We realized that information coming from a graph can be useful to better identify and highlight the connections and similarities between patients with the same disorder. Thus, in a classification problem, adding features that can help the machine learning method to identify samples belonging to the same class and better discriminate between samples belonging to different classes is extremely useful.

Concerning the feature selection algorithm the results show that in

most of the situations the best accuracy is achieved when only a small subset of features is provided to the classifier. In particular, when graph–based features are not taken in consideration, the most frequent MR markers extracted by the ranking procedure coincide with the ones mostly used by the clinicians in this kind of diagnosis. The results reported in Table 8 and in Figure 37 show that, in general, 10 or 20 features are enough to achieve the same or even better results than the ones obtained with the entire set of features. One of the most meaningful feature is, for instance, the MRPI which is considered even in (156) to be a reliable imaging measure in differentiating PSP from PD and controls subjects. These additional results related to the study of the most meaningful set of features confirm the previous results achieved in (144), where similar conclusions were obtained concerning the choice of the best features in discriminating the different disorders. When graph–based features are added to the dataset, the features ranking algorithm always places the betweenness centrality in the first positions of the ranking. In addition, for the situations in which graph–based features improve the results, except for the binary problems "MSA–P vs all" and "MSA vs all" in which the betweenness centrality is enough to improve the results, also the closeness centrality is selected and places in the first positions of the ranking. Finally, for the binary problems "PD vs all" and "PD vs PSP", even the degree is selected as one of the most meaningful features.

For what concerns future works, we first intend to apply the method developed to a larger dataset in order to test the machine learning technique on and independent evaluation test set. In addition, we believe that additional improvements could be obtained extending the current construction of the graph, in order to extract additional features. We first intend to consider different methods to build the graph by computing, for instance, different distances between the data points or evaluating correlation measures. At the same time, instead of considering all the MR markers, it would be possible to build the graph characterizing each single node by only a subset of features (maybe extracted before the computation of the graph by the ranking procedure). Finally, it could be interesting to improve the method of construction of the graph when

additional data are added to the dataset. In the current implementation, the graph is built at the beginning of the learning process using the entire dataset; thus, if two independent sets are provided, one for the training phase and the other for the test, we need to compute the graph–based features of all the data before training the classifier. As a future improvement, we could define a learning process where the graph is computed at the beginning on the training set only, and then, once the classifier is trained, a new test data can be "attached" to the graph built on the training set computing its graph–based features. In this way, we would prevent the learning process to either build the graph "a–priori" on the entire dataset, using both training and test data, or to compute it again when a data is added.

6.6 Summary

In this chapter we again applied graphs in a machine learning context. Graphs were used in a sort of pre–processing step and not as part of the algorithm considered (such as in the application of the Laplacian SVM described in the previous chapter). We modeled the dataset by means of a graph with the aim of extracting useful information to add to the original features of the samples in the dataset. Graphs have been revealed to be particularly suitable for this kind of application; various improvements were obtained.

The potentialities of graphs were studied in two different types of applications of machine learning techniques. The first one (studied and discussed in Chapter 5) revealed that graphs are extremely useful in a semi–supervised learning context, making it possible to exploit additional information from the unlabeled samples. The second one, instead, highlighted the capabilities of graphs in mining a dataset, by extracting additional information able to improve the classification performances of a learning technique applied to the original dataset without using graph–based techniques.

7 Conclusions

7.1 Concluding remarks

Graphs are extremely useful in representing and modeling data and the interactions between the different entities. They are applied in different contexts, varying from the study of the interactions between different entities in a physical/chemical system, to the study of the behavior and the dynamics of a group of agents that exchange information and opinions among each other. At the same time, a graph built on a certain dataset highlights the similarities and dissimilarities between the pairs of data, providing additional knowledge and information about the structure of a specific dataset and additional features of the data.

In **Chapter 2**, we have provided a brief description of graphs introducing the fundamental notions about graph–based features and their spectral properties used and discussed in the following chapters. In particular, we have studied graphs in two different contexts. The first one is the consensus problem, which concerns the study of the spectral properties of graphs in order to understand and model the dynamics of a group of agents that exchange their opinions, with the aim of reaching at a certain time a common opinion. The problem has been discussed in **Chapters 3** and **4**, where we have studied methods able to increase the conver-

gence rate to the consensus state. In particular, in **Chapter 3** we focused on the topology of the network of agents with the aim of determining a solution able to provide a fast convergence rate to the consensus state keeping the network as sparse as possible. This type of study reveals extremely useful in real situations characterized by huge networks with a high cost of communication between the different agents. Thus, providing a solution able, at the same time, to keep the rate of convergence to the consensus state relatively fast and to reduce the number of interconnections (hence, the communication cost) between the different agents is extremely useful. In **Chapter 4**, instead, we mainly focused on the convergence rate to the consensus state developing a method able to divide the original graph in different subgraphs (groups of agents) where their own consensus state is usually reached faster. We thus developed a sort of hierarchical method studying the spectral properties of the graph and the Cheeger's inequality obtaining satisfactory results. In almost all the examples presented, our method outperformed the classical consensus algorithm applied to the original network.

In the third part of the thesis we studied the capabilities of graphs in the machine learning context, applying them in two different classification problems. In **Chapter 5** we first continued to study the spectral properties of graphs applying a semi–supervised learning technique for the classification of flood–prone areas. In this context, we exploited the potentialities of graphs in modeling a dataset, thus its similarities and dissimilarities, with the aim of extracting useful information from unlabeled samples. In the classification problem studied, the method adopted reveals promising since better results than the ones obtained by a supervised model were achieved. Finally, the potentialities and versatility of graphs were applied in another machine learning problem. More precisely, in **Chapter 6** we modeled through a graph a dataset made of patients with different forms of Parkinsonisms, with the aim of extracting additional features able to provide useful information for the classification of the patients. In most of the situations studied the accuracy achieved adding graph–based features to the original dataset outperformed the one obtained without adding the features extracted from the graph.

In the thesis, the capabilities of graphs in studying and modeling different problems were evaluated, ranging from the study of dynamical systems to classification problems. Spectral properties of graphs revealed fundamental in the analysis of a system dynamics and in the application of a semi–supervised learning method in a classification context. At the same time, graphs where applied again in a classification problem with the aim of computing additional features from the dataset. These features revealed to be extremely useful to improve the results obtained when graph–based features were not considered.

7.2 Future directions

Several research avenues remain open in both contexts studied and discussed in the thesis. We have presented the consensus problem and we have discussed (providing theoretical and practical examples) solutions able to sparsify the original network keeping the convergence rate to the consensus state as fast as possible. We have provided a valid approach able to increase the convergence rate to the consensus state dividing the original network in many subgraphs. More detailed studies and additional developments could be carried out. More specifically, concerning the problem discussed in **Chapter 3** it would be interesting to apply techniques able to "approximate" an arbitrary graph by a sparse one as discribed in (157; 158) and compare the results with the ones achieved by our method. At the same time it would be useful to apply our method to networks derived from real situations, in particular in social contexts with individuals which interact and exchange opinions.

Concerning the hierarchical method developed in **Chapter 4**, we first intend to improve the methods developed to automatically determine sufficiently "dense" subgraphs from an original graph G. It is necessary to better study the *antenna effect* and to improve the choice of the *supernodes* when the *nearest supernode approach* is applied during the $1st$ phase of the hierarchical method. At the same time, as a future development, we intend to apply it to networks modeling real situations in order to better understand how the individuals cluster, and to automatically determine

subsets of agents with "similar" opinions.

When graphs are applied in a machine learning context, we intend to further study the application of graphs in modeling a dataset of a classification problem with the aim of extracting additional information not previously provided. This "preprocessing" step, for instance, could be applied even in the semi–supervised learning problem studied in **Chapter 5**, trying to achieve better results than the ones already obtained by the Laplacian SVM. In this way we expect, in fact, to obtain additional knowledge about the similarities and dissimilarities of the samples, which could be useful to discriminate between samples belonging to one or the other class.

In addition, we intend to develop a spectral study of the graph built on the dataset studied in **Chapter 6**. More precisely, starting from the graph modeling the original dataset, the idea is to evaluate the spectral properties of a subgraph with nodes belonging to only one class, and to study how they change if a node belonging to another class is added. Our hypothesis is that the spectral properties of subgraphs made by nodes belonging to only one class should be "better" (i.e., the graph is expected to be better clusterizable, or equivalently, the second smallest eigenvalue of the laplacian matrix should be relatively high) than the ones related to a subgraph with nodes representing samples belonging to different classes. In a sense, this is also related to the techniques used to study the consensus problem, since they have also the aim of obtaining "good" spectral properties of graph. Thus, the evaluation of the rate of change of the spectral properties of the subgraphs could be useful to better classify the data in the different classes. Note that for this kind of study it is necessary to "normalize" and adequately compare the subgraphs extracted (159; 160), since the spectral properties depend also on the dimension of the subgraphs themselves, as the studies carried out in Chapter **4** show.

Interesting and promising methods have been investigated in this work varying from the study of dynamical systems to machine learning problems. We have provided original studies and we have obtained positive results in the study of the consensus problem. At the same time we have applied a semi–supervised learning technique to a classification

problem with satisfactory results and we have added an original graph–based study to a second classification problem. Several developments and future studies remain open in both the situations considered and we believe that future improvements can be even obtained by applying techniques derived from the study of dynamical systems to a classification context.

A

Support Vector Machines

Let a set made of a finite number l of labeled training data $\{(x_i, y_i), i = 1, \ldots, l\}$ be given, with $x_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$. Here, the label -1 is used to denote the "negative" class, while $+1$ is the "positive" class label. Given a regularization parameter $\gamma_A > 0$ and a suitable function space H_K, more precisely, a reproducing kernel Hilbert space (14), the (binary) Support Vector Machine (SVM) training problem consists in searching for a classifier f^* that solves the following optimization problem: find

$$\min_{f \in H_K} \Big(\frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \gamma_A \|f\|^2_{H_K} \Big). \tag{A.1}$$

By $\|.\|^2_{H_K}$ we denote the square of the norm in the reproducing kernel Hilbert space H_K, and $(1 - y_i f(x_i))_+$ is the so–called hinge–loss function, which is defined as

$$(1 - y_i f(x_i))_+ := \max(0, 1 - y_i f(x_i)) \tag{A.2}$$

The term $\frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(x_i))_+$ in (A.1) penalizes the classification error on the training set, whereas the term $\gamma_A \|f\|^2_{H_K}$ in (A.1) enforces a small norm of the optimal solution f^* in the reproducing kernel Hilbert space H_K (i.e., typically, high smoothness for f^*). Given a (possibly unseen)

data point $x \in \mathbb{R}^m$, the optimal classifier f^* assigns to x the label $+1$ if $f^*(x) \geq 0$, otherwise it assigns to x the label -1.

The optimization problem (A.1) can be rewritten in the following way: find

$$\begin{aligned} \min_{f \in H_K, \xi_i \in \mathbb{R}} \quad & \Big(\frac{1}{l} \sum_{i=1}^{l} \xi_i + \gamma_A \|f\|^2_{H_k} \Big) \\ \text{subject to} \quad & y_i f(x_i) \geq 1 - \xi_i, \text{for } i = 1, \ldots, l\,, \\ & \xi_i \geq 0, \qquad\qquad \text{for } i = 1, \ldots, l\,. \end{aligned} \tag{A.3}$$

We denote by $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ the (uniquely determined) kernel function associated with the reproducing kernel Hilbert space H_K (14). The optimal solution f^* of the optimization problem (A.3) is provided by the Representer Theorem (14) in the following form:

$$f^*(x) = \sum_{i=1}^{l} \alpha_i^* K(x, x_i), \tag{A.4}$$

where the optimal coefficients $\alpha_i^* \in \mathbb{R}$. Therefore, solving the optimization problem (A.3) is reduced to determining the finite–dimensional coefficients α_i^* that minimize its objective, when the function f is constrained to have the form (A.4). For a reproducing kernel Hilbert space H_K, the kernel K has often a simple expression. This is the case, e.g., of the linear kernel

$$K(x, y) := \langle x, y \rangle_{\mathbb{R}^m}\,, \tag{A.5}$$

and of the Gaussian kernel

$$K(x, y) := \exp^{-\frac{\|x-y\|^2_{\mathbb{R}^m}}{2\sigma^2}}, \tag{A.6}$$

where $\sigma > 0$ is a fixed width parameter. It often happens that only a small subset of the coefficients α_i^* (with respect to their total number l) is different from 0; the input data points x_i associated with non–zero α_i^* are called support vectors. In practice, a binary SVM classifier can be interpreted as a binary linear classifier in a (possibly infinite–dimensional) auxiliary feature space associated with the reproducing kernel Hilbert

space H_K. The mapping between the original feature space $\mathbb{R}^m$ and the auxiliary feature space is typically nonlinear. A binary SVM classifier often allows one to separate data points that are not linearly separable in the original feature space.

Laplacian Support Vector Machines

Let us assume that a set made of a finite number l of labeled training data $\{(x_i, y_i), i = 1, , l\}$, with $x_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$ is available. We also assume the presence of a second set made of a finite number u of unlabeled training data $\{x_j, j = l + 1, , l + u\}$, with $x_j \in \mathbb{R}^m$. H_K denotes a reproducing kernel Hilbert space, whereas $\gamma_A > 0$ is a regularization parameter. We also assume that a second regularization parameter $\gamma_I > 0$ is given. With these premises, the (binary) Laplacian Support Vector Machine (LapSVM) (110) extends the SVM formulation described in Appendix A by solving the following optimization problem (which is inspired by the principle of manifold regularization, see Section 5.2.1): find

$$\min_{f \in H_K} \left(\frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(x_i))_+ + \gamma_A \|f\|^2_{H_K} + \frac{\gamma_I}{(u+l)^2} \mathbf{f}^T L \mathbf{f} \right) \tag{B.1}$$

where $\mathbf{f} := [f(x_1), \ldots, f(x_{l+u})]^T$, and $L \in \mathbb{R}^{(l+u)\times(l+u)}$ is the graph Laplacian matrix defined as $L := D - W$. W denotes the symmetric weighted adjacency matrix, and D is the degree matrix. Likewise in Appendix A, the goal of the term $\frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(x_i))_+$ in formula (B.1) is to penalize the classification error on the training set, whereas the term $\gamma_A \|f\|^2_{H_K}$ in (B.1) enforces a small norm of the optimal solution f^* in the

reproducing kernel Hilbert space H_K (i.e., typically, high smoothness for f^*). Finally, the term $\frac{\gamma_I}{(u+l)^2}\mathbf{f}^T L\mathbf{f}$ enforces smoothness of the optimal solution f^* also with respect to the graph approximation of the Riemannian manifold. The expression of the optimal solution f^* of problem (B.1) follows again from another form of the Representer Theorem, and it is given by

$$f^*(x) := \sum_{i=1}^{l+u} \alpha_i^* K(x, x_i), \tag{B.2}$$

for suitable optimal coefficients $\alpha_i^* \in \mathbb{R}$. Again, solving the optimization problem (B.1) is reduced to determine the finite–dimensional coefficients α_i^* that minimize its objective, when the function f has the form (B.2).

C

Additional studies of the Cheeger's inequality

In this appendix we further study the bounds on the second–smallest eigenvalue of the Laplacian matrix of particular examples of graphs. We thus continue the study presented in Section 4.6.1 in order to better understand possible situations that can arise when an original graph is divided in several subgraphs.

Like in Section 4.6.1 we are not interested in evaluating all the possible cuts that divide the original graph G in two disjoint subgraphs; we intend to consider only the cuts that partition G in two connected subgraphs. Thus, in the following studies we will provide upper bounds on the second–smallest eigenvalue of the Laplacian matrix considering only connected subgraphs.

Proposition 13. *Let $G = (V, E)$ be a chain graph with $|V| = N$ and let the transition probability matrix P associated to G computed following the procedure described in Section 4.2.1. Then, the larger the dimension of the graph, i.e., N, the smaller the Cheeger's upper bound on the second–smallest eigenvalue ξ_1 of the normalized Laplacian matrix associated to G.*

Proof. For the sake of simplicity, we investigate a chain with an even number of nodes N. The procedure followed to compute P associates a weight w to every edge of the chain, a self–loop with weight equal to

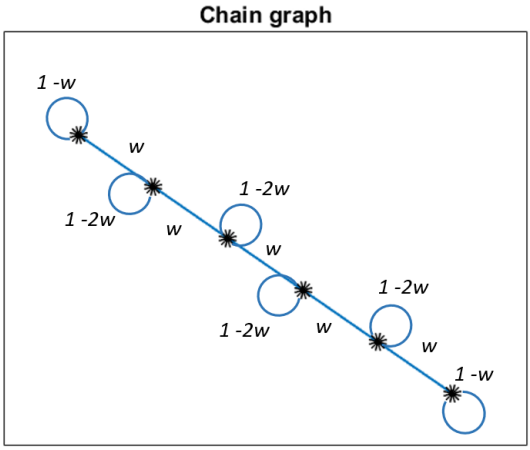

Figure 39: Chain graph with 6 nodes.

$1-w$ for the extremity nodes, and self–loops with weight equal to $1-2w$ to each "internal" node (like the graph in Figure 39). In this case, the minimum among all the possible cuts is obtained cutting the chain in the middle, obtaining two subgraphs with the same number of nodes $\frac{N}{2}$ [1]. The Cheeger's constant is then bounded from above as $\Phi(G) \leq \frac{2}{N}w$. Thus, an upper bound on the second–smallest eigenvalue of the Laplacian is given by:

$$\xi_1 \leq \frac{4w}{N}.$$

Being w constant (note that w cannot be larger than 1), it follows that the larger the dimension of the graph, i.e., the larger the value of N, the smaller the upper bound on ξ_1.

Remark 11. *Let $G = (V, E)$ be a chain graph with $|V| = N$. Now, let us consider a transition probability matrix P without self–loops assigned to the "internal" nodes. Then, the larger the dimension of the graph, i.e., N, the smaller the Cheeger's upper bound on the second–smallest eigenvalue ξ_1 of the normalized Laplacian matrix associated to G.*

[1] For an odd value of N, two cuts provide the minimum among all the possible cuts. They are the ones that divide the chain in one subgraph with $\frac{N+1}{2} - 1$ nodes and the other with $\frac{N}{2}$ nodes.

If no self–loops are assigned to the "internal nodes", then a weight $w = \frac{1}{2}$ is associated to the "internal" edges. In this particular situation, an upper bound on the ξ_1 is the following:

$$\xi_1 \leq \frac{2}{N}.$$

Again the upper bound on the Cheeger's constant depends on the value of N and the larger N the smaller the upper bound on ξ_1. □

Proposition 14. *Let $G = (V, E)$ be a graph made of two complete graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2| = N$, connected by an edge. Let P the associated transition probability matrix computed following the procedure described in Section 4.2.1; then, the larger the value of N, the smaller the Cheeger's upper bound on the second–smallest eigenvalue ξ_1 of the normalized Laplacian matrix associated to G.*

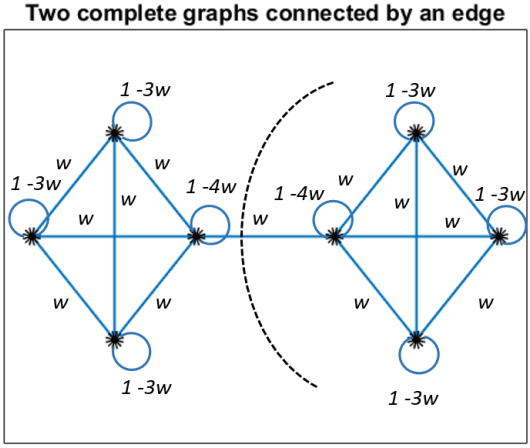

Figure 40: Two complete graphs connected by an edge.

Proof. We first consider a simple example with $N = 4$ (Figure 40). The entries of matrix P present a value equal to w for each non self–loop edge, while to the self–loops it is associated a value $1 - wd_i$, with d_i the degree of the corresponding node. Limiting to connected subgraphs, the

cut that achieves the minimum in the definition of the Cheeger's constant is the one shown in Figure 40 that divides G in the two complete graphs, with $\Phi(G) = \frac{w}{N}$, where $N = 4$. An upper bound on the second–smallest eigenvalue of the Laplacian matrix is then given by:

$$\xi_1 \leq \frac{w}{2}.$$

The smaller w the smaller the upper bound on ξ_1.

Again, if we consider a generic value for N, with the same procedure adopted to compute P, limiting to connected subgraphs, the cut that achieves the minimum in the definition of the Cheeger's constant is again the one that divides G in the two complete graphs. In particular, an upper bound on ξ_1 is the following:

$$\xi_1 \leq 2\frac{w}{N}.$$

Since $w < \frac{1}{N}$, again it follows that the larger the value of N the smaller the upper bound on ξ_1. □

In the following, we better analyze the different cuts we considered in the example with two complete graphs connected by an edge and $N = 4$ to compute an upper bound on the Cheeger's constant. We aim at showing that the cut that minimizes the ratio on the right–hand side of Formula (2.6), when one limits to consider connected subgraphs, is exactly the one reported in Figure 40. Figure 41 shows the admissible cuts that divide the original graph G in two connected subgraphs when $N = 4$. Denoting with S a generic subset of nodes of the original graph, we compute

$$\Phi(S) = \frac{|\partial(S)|}{\min\{\text{vol}(S), \text{vol}(V \setminus S)\}}$$

for every choice of $S \subset V$ and then we determine the minimum among all the subsets S such that both S and $V \setminus S$ are connected. The transition probability matrix P is again computed following the procedure described in Section 4.2.1, assigning a weight w to the non self–loop edges. Denoting with S_1 (and respectively $V \setminus S_1$) the subsets obtained partitioning G by means of **cut1** (see Figure 41), and equivalently with S_2 and $V \setminus S_2$, S_3 and $V \setminus S_3$, S_4 and $V \setminus S_4$ the other sets obtained by the other

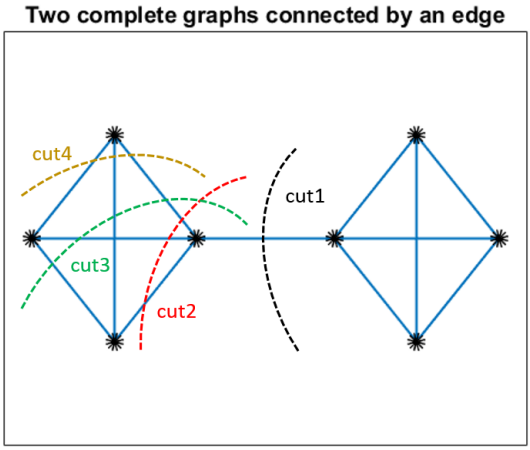

Figure 41: All the possible cuts that divide two complete graphs connected by an edge in two connected subgraphs.

cuts we have: $\Phi(S_1) = \frac{w}{N}$, $\Phi(S_2) = \frac{(N-1)w}{N-1} = w$, $\Phi(S_3) = \frac{2(N-2)w}{N-2} = 2w$, and $\Phi(S_4) = (N-1)w$. Where N in this particular example is equal to 4. Then, the smallest value of $\Phi(S)$ is obtained by dividing G in the two complete graphs with $N = 4$ nodes each. This argument can be generalized for every value of N.

References

[1] D. Cook and L. Holder, *Mining Graph Data*. Wiley, 2007. 2

[2] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Elsevier, 2011. 2

[3] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003. 2, 3

[4] A. Mashaghi, A. Ramezanpour, and V. Karimipour, "Investigation of a protein complex network," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 41, no. 1, pp. 113–121, 2004. 2

[5] T. Milenkovic, *Graph-theoretical approaches for studying biological networks*. PhD thesis, University of California, Irvine, 2008. 2

[6] M. Newman, *Networks: an introduction*. Oxford University Press, 2010. 2, 11

[7] M. S. Handcock, A. E. Raftery, and J. M. Tantrum, "Model-based clustering for social networks," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 170, no. 2, pp. 301–354, 2007. 3

[8] J. Scott, *Social network analysis*. Sage, 2012. 3

[9] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013. 3

[10] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. 3

[11] Z. Zhu, F. Liu, and D. Liao, "Rumor spreading and degree-related preference mechanism on a small-world network," *Sociology Mind*, vol. 2, no. 4, pp. 477–481, 2012. 3

[12] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007. 3

[13] P. Latouche and F. Rossi, "Graphs in machine learning: an introduction," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 207–218, 2015. 3

[14] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004. 3, 113, 115, 169, 170

[15] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proceedings of the 19th international conference on machine learning*, pp. 315–322, 2002. 3

[16] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *The Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010. 3

[17] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*, pp. 129–143, Springer, 2003. 3

[18] K. M. Borgwardt, H.-P. Kriegel, S. Vishwanathan, and N. N. Schraudolph, "Graph kernels for disease outcome prediction from protein-protein interaction networks," in *Pacific symposium on biocomputing*, vol. 12, pp. 4–15, World Scientific, 2007. 3

[19] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl 1, pp. i47–i56, 2005. 3

[20] H. Bunke, "Recent developments in graph matching," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2, pp. 117–124, IEEE, 2000. 4

[21] H. Bunke, "Graph matching: Theoretical foundations, algorithms, and applications," in *Proc. Vision Interface*, vol. 2000, pp. 82–88, 2000. 4

[22] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 1, p. 1450001, 2014. 4

[23] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 6, pp. 1048–1058, 2009. 4

[24] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 775–779, 1997. 4

[25] A. Shokoufandeh and S. Dickinson, "A unified framework for indexing and matching hierarchical shape structures," in *Visual Form 2001*, pp. 67–84, Springer, 2001. 4

[26] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1, pp. 309–320, 2000. 4

[27] A. Schenker, M. Last, H. Bunke, and A. Kandel, "Classification of web documents using a graph model," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pp. 240–244, IEEE, 2003. 4

[28] A. Dominik, Z. Walczak, and J. Wojciechowski, "Classification of web documents using a graph-based model and structural patterns," in *Knowledge Discovery in Databases: PKDD 2007*, pp. 67–78, Springer, 2007. 4

[29] L. B. Holder and D. J. Cook, "Graph-based relational learning: Current and future directions," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 90–93, 2003. 4

[30] E. Lovisari and S. Zampieri, "Performance metrics in the average consensus problem: A tutorial," *Annual Reviews in Control*, vol. 36, pp. 26–41, 2012. 4, 25, 26

[31] Q. Wang, H. Gao, F. Alsaadi, and T. Hayat, "An overview of consensus problems in constrained multi-agent coordination," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 275–284, 2014. 4

[32] R. Hegselmann and U. Krause, "Opinion dynamics and bounded confidence models, analysis, and simulation," *Journal of Artificial Societies and Social Simulation*, vol. 5, no. 3, 2002. 4

[33] R. Olfati-Saber, A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. 4

[34] V. Blondel, J. M. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *IEEE Conference on Decision and Control*, vol. 44, p. 2996, IEEE; 1998, 2005. 4

[35] A. Jadbabaie and J. Lin, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, 2003. 4

[36] M. Cao, A. Morse, and B. Anderson, "Coordination of an asynchronous multi-agent system via averaging," in *16th IFAC World Congress*, pp. 4–8, 2005. 4

[37] S. Kar and J. M. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 355–369, 2009. 4

[38] W. Ren and R. W. Beard, "Overview of consensus algorithms in cooperative control," *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*, pp. 3–22, 2008. 4

[39] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Distributed averaging on asynchronous communication networks," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 7446–7451, IEEE, 2005. 5

[40] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006. 5

[41] M. Cao, A. S. Morse, and B. D. Anderson, "Reaching a consensus in a dynamically changing environment: Convergence rates, measurement delays, and asynchronous events," *SIAM Journal on Control and Optimization*, vol. 47, no. 2, pp. 601–623, 2008. 5

[42] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 1859–1864, IEEE, 2005. 5

[43] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973. 5

[44] R. Merris, "Laplacian matrices of graphs: a survey," *Linear algebra and its applications*, vol. 197, pp. 143–176, 1994. 5

[45] M. Cao, D. A. Spielman, and A. S. Morse, "A lower bound on convergence of a distributed network consensus algorithm," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 2356–2361, IEEE, 2005. 5

[46] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009. 5

[47] H. Hao and P. Barooah, "Improving convergence rate of distributed consensus through asymmetric weights," in *American Control Conference (ACC), 2012*, pp. 787–792, IEEE, 2012. 5

[48] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM review*, vol. 46, no. 4, pp. 667–689, 2004. 5, 25, 28, 29, 43, 65, 99, 109

[49] F. R. Chung, "Lectures on spectral graph theory," *CBMS Lectures, Fresno*, 1996. 6, 17, 21

[50] R. Diestel, *Graph theory {graduate texts in mathematics; 173}*. Springer-Verlag Berlin and Heidelberg GmbH & amp, 2000. 11

[51] J. L. Gross and J. Yellen, *Handbook of graph theory*. CRC press, 2004. 11

[52] C. Godsil and G. F. Royle, *Algebraic graph theory*, vol. 207. Springer Science & Business Media, 2013. 11

[53] C. Pozrikidis, *An Introduction to Grids, Graphs, and Networks*. Oxford University Press, 2014. 11

[54] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1979. 11

[55] U. v. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, 2007. 15, 16, 18, 19, 21

[56] A. E. Brouwer and W. H. Haemers, *Spectra of graphs*. Springer Science & Business Media, 2011. 15

[57] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The laplacian spectrum of graphs," *Graph theory, combinatorics, and applications*, vol. 2, pp. 871–898, 1991. 15

[58] B. Mohar and M. Juvan, "Some applications of laplace eigenvalues of graphs," in *Graph Symmetry: Algebraic Methods and Applications*, vol. 497, pp. 227–275, 1997. 15, 28

[59] F. J. Hall, "The adjacency matrix, standard laplacian, and normalized laplacian, and some eigenvalue interlacing results," *Department of Mathematics and Statistics, Georgia State University, Atlanta*, 2010. 16

[60] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. 17, 19, 71

[61] M. C. V. Nascimento and A. C. P. L. F. Carvalho, "Spectral methods for graph clustering—a survey," *European Journal of Operational Research*, vol. 211, no. 2, pp. 221–231, 2011. 17, 19

[62] L. W. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992. 17

[63] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," *Physics reports*, vol. 424, no. 4, pp. 175–308, 2006. 17

[64] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001. 18

[65] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002. 18

[66] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *Computer-aided design of integrated circuits and systems, ieee transactions on*, vol. 11, no. 9, pp. 1074–1085, 1992. 19

[67] M. Meila and J. Shi, "A random walks view of spectral segmentation," 2001. 20

[68] G. W. Stewart, "Matrix perturbation theory," 1990. 20

[69] R. Bhatia, *Matrix analysis*, vol. 169. Springer Science & Business Media, 2013. 20

[70] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. iii," *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970. 20, 21

[71] F. R. Chung, *Spectral graph theory*, vol. 92. American Mathematical Soc., 1997. 21, 96, 97

[72] C. Castellano, S. Fortunato, and V. Loreto, "Statistical physics of social dynamics," *Reviews of modern physics*, vol. 81, no. 2, p. 591, 2009. 24

[73] S. Zampieri, "Consensus and distributed estimation." Slides of a seminar held at HYCON2 PhD School on Control of Networked and Large–Scale Systems, 2012. 24

[74] J. N. Tsitsiklis, "Problems in decentralized decision making and computation.," tech. rep., DTIC Document, 1984. 24

[75] M. J. Fischer, "The consensus problem in unreliable distributed systems (a brief survey)," in *Foundations of Computation Theory*, pp. 127–140, Springer, 1983. 25

[76] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, 2004. 26

[77] R. B. Bapat, *Graphs and matrices*. Springer, 2010. 27

[78] A. Katok and B. Hasselblatt, *Introduction to the modern theory of dynamical systems*, vol. 54. Cambridge University Press, 1997. 28

[79] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012. 28, 32

[80] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. 30

[81] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994. 31, 32

[82] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004. 31, 43, 57

[83] N. Dhingra, F. Lin, M. Fardad, and M. R. Jovanovic, "On identifying sparse representations of consensus networks," in *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems, Santa Barbara, CA*, pp. 305–310, 2012. 31, 32

[84] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011. 32

[85] H. Dym, *Linear algebra in action*, vol. 78. American Mathematical Soc., 2013. 34

[86] T. Hastie, J. Taylor, and R. Tibshirani, "Forward stagewise regression and the monotone LASSO," *Electronic Journal of Statistics*, vol. 1, pp. 1–29, 2007. 35

[87] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1996. 36

[88] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. 40

[89] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas*. Princeton University Press, 2009. 40

[90] K. C. Border, *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer-Verlag, New York, 2006. 44

[91] N. L. Stokey, *Recursive Methods in Economic Dynamics*. Harvard University Press, 1989. 46

[92] G. Rúcker and C. Rúcker, "Automatic enumeation of all connected subgraphs," *MATCH Communications in Mathematical and in Computer Chemistry*, vol. 41, pp. 145–149, 2000. 51

[93] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 51

[94] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l_1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008. 61

[95] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006. 61

[96] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 231–245, 2013. 62

[97] M. Epstein, K. Lynch, K. H. Johansson, and R. M. Murray, "Using hierarchical decomposition to speed up average consensus," in *International Federation of Automatic Control (IFAC) World Congress, Seoul, Korea, July 2008*, pp. 612–618, 2008. 65, 80

[98] X. Li and H. Bai, "Consensus on hierarchically decomposed topology to accelerate convergence and to improve delay robustness," in *Control and Decision Conference (CCDC), 2012 24th Chinese*, pp. 597–602, IEEE, 2012. 65

[99] F. Fagnani, "Consensus dynamics over networks," *Technical Paper*, 2014. 66

[100] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," *Lecture Notes in Control and Information Sciences*, vol. Networked control systems, A. Bemporad, M. Heemels, M. Johansson Eds. 67

[101] P. Erdos and A. Renyi, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959. 83

[102] B. Bollobás, *Random graphs*. Springer, 1998. 83

[103] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999. 83

[104] A.-L. Barabási and E. Bonabeau, "Scale-free networks," *Scientific American*, vol. 288, no. 5, pp. 50–59, 2003. 83

[105] O. Hein, M. Schwind, and W. König, "Scale-free networks," *Wirtschaftsinformatik*, vol. 48, no. 4, pp. 267–275, 2006. 83

[106] A.-L. Barabási, "Scale-free networks: a decade and beyond," *science*, vol. 325, no. 5939, p. 412, 2009. 83

[107] E. Mossel, J. Neeman, and A. Sly, "Stochastic block models and reconstruction," *arXiv preprint arXiv:1202.1499*, 2012. 88

[108] E. Mossel, J. Neeman, and A. Sly, "Reconstruction and estimation in the planted partition model," *Probability Theory and Related Fields*, pp. 1–31, 2014. 88

[109] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002. 90

[110] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of machine Learning Research*. 111, 113, 172

[111] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009. 111, 120

[112] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 4 ed., 2009. 113, 124, 150

[113] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006. 113, 124

[114] M. P. Do Carmo, *Differential geometry of curves and surfaces*, vol. 2. Prentice-hall Englewood Cliffs, 1976. 113

[115] F. Guzzetti, C. P. Stark, and P. Salvati, "Evaluation of flood and landslide risk to the population of italy," *Environmental Management*, vol. 36, no. 1, pp. 15–36, 2005. 116

[116] M. Horritt and P. Bates, "Evaluation of 1d and 2d numerical models for predicting river flood inundation," *Journal of Hydrology*, vol. 268, no. 1, pp. 87–99, 2002. 116

[117] N. M. Hunter, P. D. Bates, M. S. Horritt, and M. D. Wilson, "Simple spatially-distributed models for predicting flood inundation: a review," *Geomorphology*, vol. 90, no. 3, pp. 208–225, 2007. 116

[118] P. Bates, K. Marks, and M. Horritt, "Optimal use of high-resolution topographic data in flood inundation models," *Hydrological Processes*, vol. 17, no. 3, pp. 537–557, 2003. 116

[119] J. C. Gallant and T. I. Dowling, "A multiresolution index of valley bottom flatness for mapping depositional areas," *Water Resources Research*, vol. 39, no. 12, 2003. 116

[120] F. Giannoni, G. Roth, and R. Rudari, "A procedure for drainage network identification from geomorphology and its application to the prediction of the hydrologic response," *Advances in Water Resources*, vol. 28, no. 6, pp. 567–581, 2005. 116

[121] B. Dodov and E. Foufoula-Georgiou, "Floodplain morphometry extraction from a high-resolution digital elevation model: a simple algorithm for regional analysis studies," *Geoscience and Remote Sensing Letters, IEEE*, vol. 3, no. 3, pp. 410–413, 2006. 116

[122] F. Nardi, S. Grimaldi, M. Santini, A. Petroselli, and L. Ubertini, "Hydrogeomorphic properties of simulated drainage patterns using digital elevation models: the flat area issue/propriétés hydro-géomorphologiques de réseaux de drainage simulés à partir de modèles numériques de terrain: la question des zones planes," *Hydrological Sciences Journal*, vol. 53, no. 6, pp. 1176–1193, 2008. 116

[123] M. Santini, S. Grimaldi, F. Nardi, A. Petroselli, and M. C. Rulli, "Preprocessing algorithms and landslide modelling on remotely sensed dems," *Geomorphology*, vol. 113, no. 1, pp. 110–125, 2009. 116

[124] N. S. Noman, E. J. Nelson, and A. K. Zundel, "Review of automated floodplain delineation from digital terrain models," *Journal of Water Resources Planning and Management*, vol. 127, no. 6, pp. 394–402, 2001. 116

[125] F. Nardi, E. R. Vivoni, and S. Grimaldi, "Investigating a floodplain scaling relation using a hydrogeomorphic delineation method," *Water Resources Research*, vol. 42, no. 9, 2006. 116

[126] S. Manfreda, M. Di Leo, and A. Sole, "Detection of flood-prone areas using digital elevation models," *Journal of Hydrologic Engineering*, vol. 16, no. 10, pp. 781–790, 2011. 116

[127] M. Degiorgis, G. Gnecco, S. Gorni, G. Roth, M. Sanguineti, and A. C. Taramasso, "Classifiers for the detection of flood–prone areas using remote sensed elevation data," *Journal of Hydrology*, vol. 470–471, pp. 302–315, 2012. 116, 117, 118, 137

[128] M. Degiorgis, G. Gnecco, S. Gorni, G. Roth, M. Sanguineti, and A. C. Taramasso, "Flood hazard assessment via threshold binary classifiers: case study of the tanaro river basin," *Irrigation and Drainage*, vol. 62, no. S2, pp. 1–10, 2013. 116, 117, 118

[129] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *The Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, 2012. 126

[130] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006. 133

[131] L. L. Rodríguez, S. Vera, F. Pérez, N. Lanconelli, R. Morisi, B. Donini, D. Turco, C. Corsi, C. Lamberti, G. Gavidia, M. Bordone, E. Soudah, N. Curzen, J. Rosengarten, J. M. Morgan, J. Herrero, and M. Á. G. Ballester, "Supervised learning modelization and segmentation of cardiac scar in delayed enhanced MRI," in *Statistical Atlases and Computational Models of the Heart. Imaging and Modelling Challenges*, pp. 53–61, Springer, 2012. 139

[132] R. Morisi, B. Donini, N. Lanconelli, J. Rosengardern, J. Morgan, S. Harden, and N. Curzen, "Semi–automated scar detection in delayed enhanced cardiac magnetic resonance images," *International Journal of Modern Physics C*, vol. 26, no. 1, 2015. 139

[133] C. Rusu, R. Morisi, D. Boschetto, R. Dharmakumar, and S. A. Tsaftaris, "Synthetic generation of myocardial blood-oxygen-level-dependent mri time series via structural sparse decomposition modeling," *IEEE Transactions on Medical Imaging*, vol. 33, no. 7, pp. 1422–1433, 2014. 139

[134] S. Sharma, C. S. Moon, A. Khogali, A. Haidous, A. Chabenne, C. Ojo, M. Jelebinkov, Y. Kurdi, and M. Ebadi, "Biomarkers in parkinson's disease (recent update)," *Neurochemistry international*, vol. 63, no. 3, pp. 201–229, 2013. 140

[135] P. Mahlknecht, A. Hotter, A. Hussl, R. Esterhammer, M. Schocke, and K. Seppi, "Significance of mri in diagnosis and differential diagnosis of parkinson's disease," *Neurodegenerative Diseases*, vol. 7, no. 5, pp. 300–318, 2010. 140

[136] J. Prodoehl, H. Li, P. J. Planetta, C. G. Goetz, K. M. Shannon, R. Tangonan, C. L. Comella, T. Simuni, X. J. Zhou, S. Leurgans, D. M. Corcos, and D. E. Vallancourt, "Diffusion tensor imaging of parkinson's disease, atypical parkinsonism, and essential tremor," *Movement Disorders*, vol. 28, no. 13, pp. 1816–1822, 2013. 140

[137] G. Öz, "Mr spectroscopy: A longitudinal biomarker for substantia nigra pathology in parkinson's disease?," *Movement Disorders*, vol. 30, no. 10, pp. 1304–1305, 2015. 140

[138] N. Seraji-Bozorgzad, F. Bao, E. George, S. Krstevska, V. Gorden, J. Chorostecki, C. Santiago, I. Zak, C. Caon, and O. Khan, "Longitudinal study of the substantia nigra in parkinson disease: A high-field 1h-mr spectroscopy imaging study," *Movement Disorders*, vol. 30, no. 10, pp. 1400–1404, 2015. 140

[139] A. Cherubini, M. Morelli, R. Nisticó, M. Salsone, G. Arabia, R. Vasta, A. Augimeri, M. E. Caligiuri, and A. Quattrone, "Magnetic resonance support vector machine discriminates between parkinson disease and progressive supranuclear palsy," *Movement Disorders*, vol. 29, no. 2, pp. 266–269, 2014. 140, 160

[140] D. Long, J. Wang, M. Xuan, Q. Gu, X. Xu, D. Kong, and M. Zhang, "Automatic classification of early parkinson's disease with multi-modal mr imaging," *PLoS ONE*, vol. 7, no. 11, p. e47714, 2012. 140, 160

[141] S. Haller, S. Badoud, D. Nguyen, V. Garibotto, K. Lovblad, and P. Burkhard, "Individual detection of patients with parkinson disease using support vector machine analysis of diffusion tensor imaging data: initial results," *American Journal of Neuroradiology*, vol. 33, no. 11, pp. 2123–2128, 2012. 140, 160

[142] A. Cherubini, R. Nisticó, F. Novellino, M. Salsone, S. Nigro, G. Donzuso, and A. Quattrone, "Magnetic resonance support vector machine discriminates essential tremor with rest tremor from tremor-dominant parkinson disease," *Movement Disorders*, vol. 29, no. 9, pp. 1216–1219, 2014. 141

[143] A. F. Marquand, M. Filippone, J. Ashburner, M. Girolami, J. Mourao-Miranda, G. J. Barker, S. C. Williams, P. N. Leigh, and C. R. Blain, "Automated, high accuracy classification of parkinsonian disorders: a pattern recognition approach," *PloS one*, vol. 8, no. 7, p. e69237, 2013. 141, 161

[144] R. Morisi, G. Gnecco, N. Lanconelli, S. Zanigni, D. N. Manners, C. Testa, S. Evangelisti, L. L. Gramegna, C. Bianchini, P. Cortelli, C. Tonon, and R. Lodi, "Binary and multi-class parkinsonian disorders classification using support vector machines," in *Pattern Recognition and Image Analysis*, pp. 379–386, Springer, 2015. 141, 144, 161, 162

[145] D. J. Gelb, E. Oliver, and S. Gilman, "Diagnostic criteria for parkinson disease," *Archives of neurology*, vol. 56, no. 1, pp. 33–39, 1999. 142

[146] I. Litvan, Y. Agid, J. Jankovic, C. Goetz, J. P. Brandel, E. C. Lai, G. Wenning, L. D'Olhaberriague, M. Verny, K. R. Chaudhuri, and A. McKee, "Accuracy of clinical criteria for the diagnosis of progressive supranuclear palsy (steele-richardson-olszewski syndrome)," *Neurology*, vol. 46, no. 4, pp. 922–930, 1996. 142

[147] A. Quattrone, G. Nicoletti, D. Messina, F. Fera, F. Condino, P. Pugliese, P. Lanza, P. Barone, L. Morgante, M. Zappia, and U. Aguglia, "Mr imaging index for differentiation of progressive supranuclear palsy from parkinson disease and the parkinson variant of multiple system atrophy 1," *Radiology*, vol. 246, no. 1, pp. 214–221, 2008. 143

[148] G. Rizzo, P. Martinelli, D. Manners, C. Scaglione, C. Tonon, P. Cortelli, E. Malucelli, S. Capellari, C. Testa, P. Parchi, P. Montagna, B. Barbiroli, and R. Lodi, "Diffusion-weighted brain imaging study of patients with clinical diagnosis of corticobasal degeneration, progressive supranuclear palsy and parkinson's disease," *Brain*, vol. 131, no. 10, pp. 2690–2700, 2008. 143

[149] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data mining techniques for the life sciences*, pp. 223–239, Springer, 2010. 146

[150] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011. 147, 150

[151] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999. 147

[152] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *Neural Networks, IEEE Transactions on*, vol. 5, no. 4, pp. 537–550, 1994. 150

[153] S. H. Park, J. M. Goo, and C.-H. Jo, "Receiver operating characteristic (roc) curve: practical review for radiologists," *Korean Journal of Radiology*, vol. 5, no. 1, pp. 11–18, 2004. 150

[154] D. G. Altman and J. M. Bland, "Diagnostic tests 1: Sensitivity and specificity.," *BMJ: British Medical Journal*, vol. 308, no. 6943, p. 1552, 1994. 155

[155] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson, "Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach," *Biometrics*, pp. 837–845, 1988. 161

[156] M. Morelli, G. Arabia, D. Messina, B. Vescio, M. Salsone, C. Chiriaco, P. Perrotta, F. Rocca, G. L. Cascini, G. Barbagallo, S. Nigro, and A. Quattrone, "Effect of aging on magnetic resonance measures differentiating progressive supranuclear palsy from parkinson's disease," *Movement Disorders*, vol. 29, no. 4, pp. 488–495, 2014. 162

[157] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011. 166

[158] J. Batson, D. A. Spielman, N. Srivastava, and S. H. Teng, "Spectral sparsification of graphs: Theory and algorithms," *Commun. ACM*, vol. 56, no. 8, pp. 87–94, 2013. 166

[159] W. J. Lee and R. P. Duin, "An inexact graph comparison approach in joint eigenspace," in *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 35–44, Springer, 2008. 167

[160] A. Shrivastava and P. Li, "A new space for comparing graphs," in *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pp. 62–71, IEEE, 2014. 167