Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

EXPLORATORY DATA ANALYSIS USING NETWORK BASED TECHNIQUES

Clara Granell Martorell

Advisor/s: Alex Arenas Moreno and Sergio Gómez Jiménez

1st September 2012

Exploratory data analysis using network based techniques

Clara Granell Martorell

Exploratory data analysis using network based techniques

Master Thesis
by Clara Granell Martorell
for the Master Degree in Artificial Intelligence

Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
June 2012

Al Joan

Acknowledgements

I would like to thank all the people that, during the last years, have made possible the realization of this work.

Contents

List of Figures

Preface

The aim of this document is to present the work done during the development of my master thesis. The work belongs to the field of complex networks, more concretely to the detection of communities in complex networks. Chapter 1 will be an introduction of the basic concepts and motivations of this work, mainly clarifying the fields of exploratory data analysis, data clustering and complex networks. As all the work is about the finding of communities in complex networks, Chapter 2 is devoted to explain the concepts of mesoscopic structure of networks and its importance in the analysis of real networks, along with the explanations of some of the most well-known techniques to perform this analysis. All the progress done during the master thesis relies on a method for detecting communities developed in the past years by the research group I belong to. This method is known as the AFG algorithm, named after the three authors Arenas, Fernández and Gómez, and it is explained in section 2.5.2 with special emphasis. The work that I have developed is composed of two separate problems: the first one consists in designing an application to make possible the use of the AFG community detection method to perform data clustering over real world multidimensional datasets, which is explained in Chapter 3. The second work consists in improving the AFG method to make possible the detection of communities even when the difference of sizes of the communities make their detection impossible for other community detection algorithms, which can be found in Chapter 4. Chapter 5 contains the conclusions and the future lines of research derived from the present work, and in the Appendix there is a list of publications that sustain the contents presented in this document.

Chapter 1

Introduction

In this chapter we present an overview of the area of study in which this work is enclosed. As stated in the Preface, the work is mainly focused on the applications and improvements of a particular community detection method in complex networks. For the applications side, as the purpose is to adapt this method to perform in general exploratory data analysis and in particular data clustering, these concepts are overviewed in the first two sections of this chapter. Next, there is a brief introduction to complex networks that will help readers unfamiliar with this research field to understand its purposes as well as some basic concepts that will be used throughout the rest of the document.

1.1 Exploratory Data Analysis: A General Framework

The largest representation of our world is written by data, usually digital data. The analysis of these data is the key to understand our world better. Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of highlighting useful information, suggesting conclusions, and supporting decision making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains. There are many techniques to perform data analysis, using previous knowledge about them, formulating hypothesis, or in the most unfavorable case -but nonetheless the most common- without any previous knowledge about the data relations. This last situation is usually tackled using *exploratory data analysis* as an approach to analyze data sets to summarize their main characteristics in an easy-to-understand form, often with visual graphs, without using a statistical model or having formulated a hypothesis. One of the most commonly used techniques in the process of exploratory data analysis consist on the unsupervised clustering of data. We revise these concept on the next section.

1.2 Unsupervised Data Clustering

The problem of unsupervised data clustering consists in classifying elements so that elements belonging to the same cluster are more similar between them than with elements in a different cluster. An element, or pattern, is a vector of features (usually understood as a point in a multidimensional space) that describes the item we wish to classify. The goal of the process of data clustering is to organize these patterns finding a partition of the sample according to the natural classes that are present in it. Data clustering has been the subject of interest in many disciplines where the mining of raw information is crucial to understand some phenomenon or gain insight into a system. It has applications in several fields such as pattern recognition, astronomic classification, biological taxonomy, marketing, and more [17].

The process to obtain the clusters from the raw data is based on three steps: the representation of the data, the calculation of the similarity and the grouping algorithm [24]. In the first step, a representation of the patterns has to be chosen, which will normally be a multi-dimensional vector. There are no theoretical guidelines that suggest the appropriate patterns and features to use in a specific situation. Indeed, the pattern generation process is often not directly controllable; the user's role in the pattern representation process is to gather facts and conjectures about the data, optionally perform feature selection and extraction. Feature selection means choosing, from all the available features, those that will make easier the process of clustering, leaving the redundant, correlated and less informative features out of the analysis. On the other hand, feature extraction consists in transforming the original data set to a new one containing only the most relevant information. A careful investigation of the available features and any available transformations (even simple ones) can provide significantly improved clustering results. A good pattern representation can often yield a simple and easily understood clustering; a poor pattern representation may yield a complex clustering whose true structure is difficult or impossible to discern.

The second step of the process of data clustering is the definition of a measure of similarity to be applied over the set of patterns. Because of the variety of feature types and scales, the distance measure (or measures) must be chosen carefully. It is most common to calculate the dissimilarity between two patterns using a distance measure defined on the feature space, such as Euclidean distance, although there are plenty of well known distance measures in the literature, see

Distance measures	Form
Minkowski distance	$D_{ij} = \left(\sum_{l=1}^{d} \lvert x_{il} - x_{jl} \rvert^n \right)^{\frac{1}{n}}$
Euclidean distance	$D_{ij} = \left(\sum_{l=1}^{d} \lvert x_{il} - x_{jl} \rvert^2 \right)^{\frac{1}{2}}$
Manhattan distance	$D_{ij} = \sum_{l=1}^{d} \lvert x_{il} - x_{jl} \rvert$
Sup distance	$D_{ij} = \max_{1 \le l \le d} \sum_{l=1}^{d} \lvert x_{il} - x_{jl} \rvert$
Pearson correlation	$D_{ij} = \frac{(1 - r_{ij})}{2}$, where $r_{ij} = \frac{\sum_{l=1}^{d} (x_{il} - \overline{x_i})(x_{jl} - \overline{x_j})}{\sqrt{\sum_{l=1}^{d} (x_{il} - \overline{x_i})^2 \sum_{l=1}^{d} (x_{jl} - \overline{x_j})^2}}$
Mahalanobis distance	$D_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)$, where $\mathbf{S}$ is the within-group covariance matrix.
Cosine similarity	$S_{ij} = cos\alpha = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\lvert\lvert \mathbf{x}_i \rvert\rvert \lvert\lvert \mathbf{x}_j \rvert\rvert}$

Table 1.1: Most common distance measures as proposed in [47].

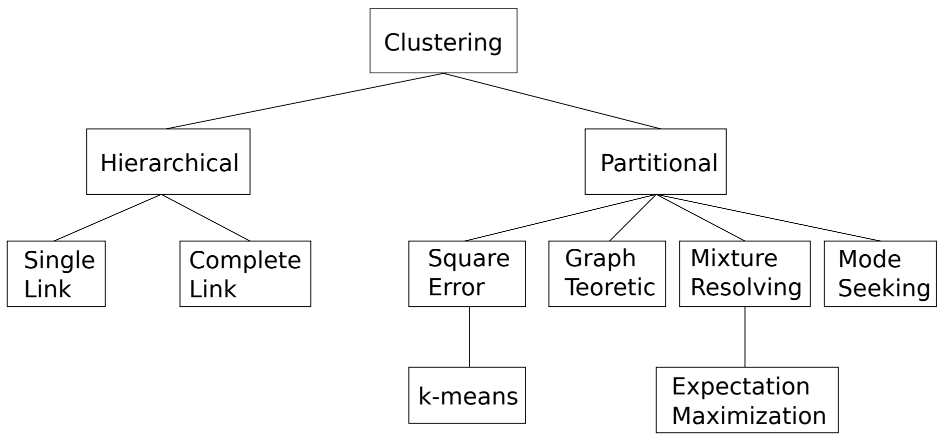

Figure 1.1: Taxonomy of the clustering algorithms as proposed in [23].

Table 1.1. This step is necessary in clustering algorithms that work on a matrix of proximity or distance values instead of working on the original pattern set. It is then useful in such situations to precompute the $\frac{n(n-1)}{2}$ pairwise distance values for the n patterns and store them in a symmetric matrix. Using the mapping to complex networks this matrix can be understood as a graph, where each node is a pattern and the links are the similarities or dissimilarities between them. Finally, the main step of the process, is the grouping (or clustering) algorithm whose aim is to decompose the similarity matrix and return the groups of data. There are a variety of clustering algorithms and each is based on different principles. In Fig. 1.1 there is a classification of clustering algorithms based on a classification proposed in [23].

The problem of clustering is inherently ill-posed, i.e. any data set can be clustered in drastically different ways, with no clear criterion for preferring one clustering over another. In particular, in the case of unsupervised approaches, a satisfactory clustering of data depends on the desired resolution which determines the number of clusters and their size. For example, k-means clustering fixes a priori the number of groups (k), which implies indeed a certain resolution of the clustering. Other algorithms such as hierarchical clustering [27] group the patterns extending the measure of distance between them to distances between clusters of patterns. This process generates a complete dendrogram. Cutting the dendrogram at different heights we obtain different partitions of the data, all them hierarchically nested. In this situation the following question arises: at what resolution should one look at the data to find a scientific meaning in the classification? We claim that the answer to this question is totally dependent on the final purpose of the classification process, and that the concept of best solution should be reconsidered. Different partitions will be representative of properties of the data at different scales and then all of them are worth to be studied.

1.3 Brief Introduction to Complex Networks

Complex networks science has been conformed as a discipline for the study of graphs representing real systems. It is an interdisciplinary field borrowing knowledge and techniques from graph theory, artificial intelligence and statistical physics. The term complex network was coined during the analysis of the power grid electricity distribution network and the Internet, to differentiate these networks from regular lattices, and perhaps more importantly, to differentiate its study from the conventional mathematical techniques used in graph theory.

Indeed, complex networks were largely studied by social scientists in the early fifties, when representing the relations between humans via social networks [44], although the analysis techniques were relatively scarce. Graph theory was mathematically grounded during the sixties [9, 13, 14] and provided the qualitative change needed for the accurate analysis of networks. Forty years later, at the beginning of this century, the physicist community became specially interested in this field and started using the common tools of statistical physics for their study [2, 7, 31, 45]. From then on, we refer to the theory of complex networks using this latter approach for their description and analysis.

Recent developments on Information and Communication Technologies have allowed the analysis of many systems that can be represented as networks, where the nodes, or vertices, represent the elements of the system, and links represent the interactions between them. In this way, systems so diverse as the WWW or Internet [22, 37], cellular metabolisms [25], or the world trade network [42], for example, have been rigorously analyzed.

There are at least three topological properties that are common to many real networks, and that differentiate these from regular lattices, or grids, namely: i) the small-world property, ii) divergent fluctuations in the number of neighbors per node (or degree) and iii) an unexpectedly large number of transitive relations, i.e. triangles, at different scales of observation of the network. Here we present a brief discussion about these properties.

1.3.1 Small-world property

From a mathematical point of view, this property can be stated as follows: the average number of steps l (from node to node) on a network of N nodes, between

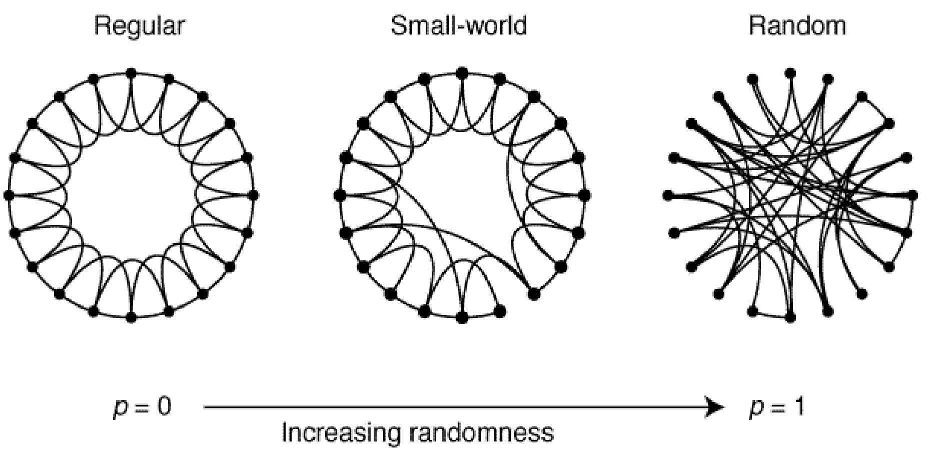

Figure 1.2: Small-world model as proposed in [46], with n=20 vertices and k=4 nearest neighbors.

any two nodes chosen at random, scales as

$$\ell \propto (\log N)^{\mu}. \tag{1.1}$$

Comparing this scaling law with the law obtained for a regular lattice of dimension d:

$$\ell \propto (N)^{1/d} \tag{1.2}$$

we can conclude that the small-world property is the responsible for the fast spreading of any type of reaction-diffusion processes that can take place on the network, as epidemic spreading, information diffusion, etc.

The first model that represents the construction of a small-world network was proposed by Watts and Strogatz [46]. It works as follows: we start with a ring of n vertices, each connected to its k nearest neighbors by undirected edges. We choose a vertex and the edge that connects it to its nearest neighbor in a clockwise sense. With probability p, we reconnect this edge to a vertex chosen uniformly at random over the entire ring, with duplicate edges forbidden; otherwise we leave the edge in place. We repeat this process by moving clockwise around the ring, considering each vertex in turn until one lap is completed. Next, we consider the edges that connect vertices to their second-nearest neighbors clockwise. As before, we randomly rewire each of these edges with probability p, and continue this process, circulating around the ring and proceeding outwards to more distant neighbors after each lap, until each edge in the original lattice has been considered once. (As there are $nk/2$ edges in the entire graph, the rewiring process stops after $k/2$ laps.) Three realizations of this process are shown, for different values of p, see Fig. 1.2. For $p = 0$, the original ring is unchanged; as p increases, the graph becomes increasingly disordered, until for $p = 1$, where all edges have been randomly rewired. The main result is that for intermediate values of p, the graph is a small-world network: highly clustered like a regular graph, yet with small characteristic path length, like a random graph.

1.3.2 Scale-free degree distribution

Many networks present clear fluctuations in the number of neighbors that different nodes have, something that has been, generally speaking, known as scale-free networks. Strictly speaking, a network is scale-free when the distribution of the number of neighbors per node (called degree k) follows a power law of the type:

$$P(k) \propto k^{-\gamma} \tag{1.3}$$

where the exponent of the distribution in most real networks is observed to be $\gamma \in (2, 3]$. Note that in this case the variance of the distribution diverges when the number of nodes grows up. This feature implies that the network is highly heterogeneous in degree and that there exists a finite probability of finding nodes with very high degree whereas most of them have low degree. Nodes with significantly high degree are called hubs.

The discovery of many real networks whose degree distribution is scale-free [4] is one of the most prominent observations of the field. This distribution has deep implications on the robustness of the topology (removing hubs would be catastrophic) and on the persistence of epidemic infections which again, due to the existence of hubs, make infections endemic.

1.3.3 Clustering and modular structure

The third property commonly observed in complex networks is the topological transitivity, usually called *clustering*. Because of the unfortunate choice of names, we have to be cautious to not mix the definition of clustering in complex networks with the problem of data clustering stated in the above section.

Clustering is a measure of the presence of triangles in the network. In social networks it is exemplified by the sentence, *the friends of my friends are my friends*, outside this scope the explanation is not so straightforward. Mathematically one of the measures of the clustering coefficient is expressed as follows:

$$C = \frac{\text{Number of closed triplets}}{\text{Total number of triplets of vertices}} \tag{1.4}$$

Clustering is one of the main elements of the complexity of networks, they are essentially the building blocks of groups of nodes highly connected between them with respect to the rest of the nodes of the network. The resulting supra-structure of the high clustering in networks are what is called modular, or community structure. Community structure will be presented in deep in the next section, being this property the central object of our approach in this thesis.

Chapter 2

Mesoscopic Structure of Networks

2.1 Community Structure

The first problem encountered when working in community detection problems is the definition of community itself. We will see that various definitions of community coexist in the literature, and while there is not a universally accepted definition, all of them share the same idea. We will first present some real world networks that have community structure and will try to unveil an intuitive definition. Afterwards, we will revise the definitions existing in the literature.

2.1.1 Real World Networks with Community Structure

In this section we shall present some examples of real networks with community structure. In this way we will see how communities look like and why they are important to understand the structure and functionality of complex networks. Social networks are paradigmatic examples of graphs with communities. The word community itself refers to a social context. People naturally tend to form groups, within their work environment, family, friends, defining communities of people. Let us show an example of social network, the Zachary's karate club, Fig. 2.1. It is a network representing the members of a karate club in the United States [48], a well-known graph regularly used as a benchmark to test community detection algorithms. It consists of 34 vertices, which are the members of this karate club, who were observed during a period of three years. Edges connect individuals who interacted outside the activities of the club. At some point, a conflict between the club president and the instructor led to the fission of

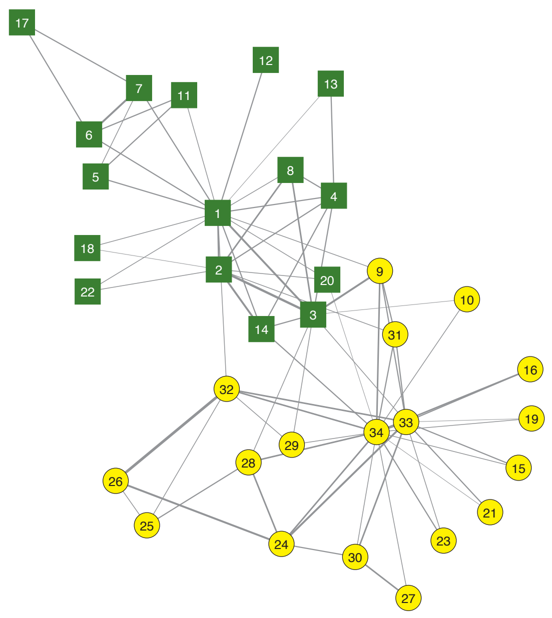

Figure 2.1: Representation of the social network corresponding to the Zachary Karate Club, a standard benchmark in community detection. Numbers correspond to different people, and triangles and squares, denote the two different groups in which the network splits. The challenge for a community detection algorithm is to unveil these groups from the original graph.

the club into two separate groups, supporting the instructor and the president, respectively. The question is whether from the original network structure it is possible to infer the composition of the two groups. Indeed, by looking at the figure one can distinguish two aggregations, one around vertices 33 and 34 (34 is the president), the other around vertex 1 (the instructor). One can also identify several vertices lying between the two main structures, like 3, 9, 10; such vertices are often misclassified by community detection methods. The original network is weighted, that is, their links have a certain weight representing the strength of the interaction between two people, however, many scientists have worked with the unweighted version of the network, something that makes more difficult the correct assignment of members to the groups they belong.

Another example of real networks are found in biology, the protein-protein interaction (PPI) networks. PPI networks are a subject of intense research in biology and bioinformatics, as the interactions between proteins are crucial to understand each metabolic process of the cell. Fig. 2.2 illustrates the PPI network of the rat proteome [26]. Each interaction (link) is derived by comparing experimentally observed interactions in other organisms. Communities (or modules) in this case correspond to functional groups, i.e. to proteins having the same or similar functions, which are expected to be involved in the same processes. Some communities are associated to cancer and metastasis processes, which indirectly shows how important detecting modules in PPI networks is.

The two networks presented have in common that they have reciprocal edges between nodes. But relationships or interactions between elements of a system do not have to be necessarily reciprocal. In many cases they have direction, that have to be taken into account to understand the system as a whole. For example, the networks representing the predator-prey models in ecosystems are one of this kind. In this kind of network, the elements of the graph represent the animals, and a directed link between them means that one of them is predator of the other. Another example, this time taken from technology, is the World Wide Web, which can be seen as a graph by representing web pages as vertices and the hyperlinks that make users move from one page to another as edges. This is also a case of directed graph, as hyperlinks are directed: if there is a link that takes the user from web page A to B, normally there is not a link that goes back from B to A. In fact, only 10% of the hyperlinks are reciprocal. This feature of real networks should be taken into account when designing community detection methods. Edge directness is not the only complication to deal with when facing the problem of community detection. In many real networks vertices may belong to more than one group, a phenomena which is known as overlapping communities. As we can see, real networks represent the complicate interconnections and relations between elements in real world, and therefore they are as complicated as the real world can be. In the end, we want to be able to extract relevant and useful information from real networks, and therefore the community detection methods have to support as many real-world features as possible.

2.1.2 Basic Definitions of Communities

The problem of community structure, intuitive at first sight, is actually ill-posed. The main elements of the problem themselves, i.e. the concepts of community and partition, are not rigorously defined, and require some degree of arbitrariness and/or common sense. Indeed, some ambiguities are hidden and there are often many equally legitimate ways of resolving them. Therefore, it is not surprising that there are plenty of recipes in the literature and that scientists do not even try to ground the problem on shared definitions.

First we have to look for a quantitative definition of community. No definition is universally accepted. As a matter of fact, the definition often depends on the specific system at hand and/or application one has in mind. From intuition and the examples of the previous section we get the notion that there must be more edges inside the community than edges linking vertices of the community with the

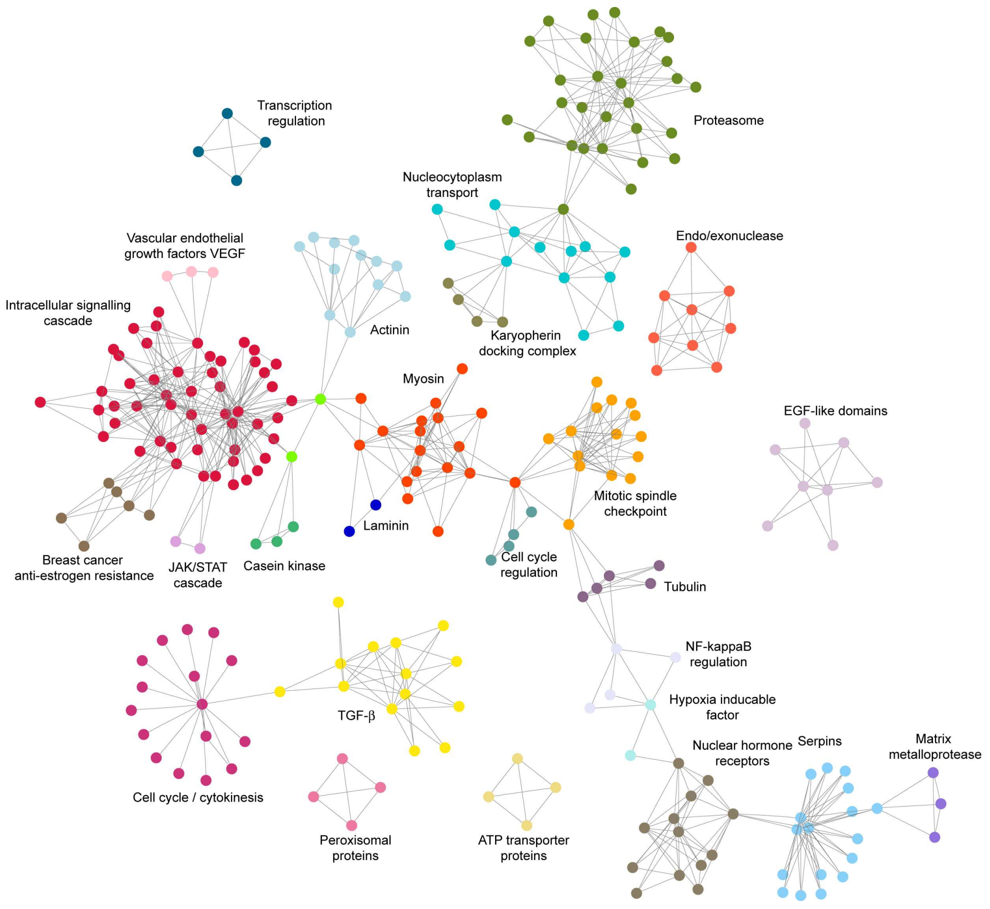

Figure 2.2: Representation of a protein-protein interaction network corresponding to the rat proteome. The links have been derived from many experimental observations in different organisms. The colors correspond to well-known functional groups according to biological knowledge of the system. The question is whether or not they also correspond to topological communities.

rest of the graph. This is the reference guideline at the basis of most community definitions. But many alternative recipes are compatible with it. Moreover, in most cases, communities are algorithmically defined, i.e. they are just the final product of the algorithm, without a precise *a priori* definition.

Let us put some notation. For a network that is undirected and has no self-loops: start with a subgraph $\mathcal{C}$ of a graph $\mathcal{G}$, with $|\mathcal{C}| = n_c$ the number of nodes in this community, and $|\mathcal{G}| = n$ the number of nodes. We define the internal and external degree of vertex $v \in \mathcal{C}$ as k_v^{int} and k_v^{ext}, which are the number of edges connecting v to other vertices of C or to the rest of the graph, respectively. The internal degree $k_{\mathcal{C}}^{int}$ of $\mathcal{C}$ is the sum of the internal degrees of its vertices. The external degree $k_{\mathcal{C}}^{ext}$ is then the sum of the external degrees of its vertices. We also have to define the intra-cluster density $\rho^{int}(\mathcal{C})$, which is the ratio between the number of internal edges of $\mathcal{C}$ and the number of all possible internal edges:

$$\rho^{int}(\mathcal{C}) = \frac{\#\text{ internal edges of }\mathcal{C}}{n_c(n_c-1)/2} \tag{2.1}$$

Likewise, the inter-cluster density $\rho^{ext}(\mathcal{C})$ is defined as the ratio between the number of edges running from the vertices of $\mathcal{C}$ to the rest of the graph and the maximum number of inter-cluster possible edges:

$$\rho^{ext}(\mathcal{C}) = \frac{\#\text{ inter-cluster edges of }\mathcal{C}}{n_c(n-n_c)} \tag{2.2}$$

For the subgraph $\mathcal{C}$ to be a community, we expect the intra-cluster density $\rho^{int}(\mathcal{C})$ to be appreciably larger than the average link density of $\mathcal{G}$. Similarly, the inter-cluster density $\rho^{ext}(\mathcal{C})$ is expected to be considerably lower than the the average link density. The goal in most of the community detection algorithms is precisely to find the best tradeoff between a large $\rho^{int}(\mathcal{C})$ and a low $\rho^{ext}(\mathcal{C})$. Another required property for $\mathcal{C}$ to be a community is connectedness. The existence of a path between each pair of the vertices in $\mathcal{C}$, running only through vertices of $\mathcal{C}$, is expected.

With these basic requirements clarified, we will now briefly introduce the two main definitions of community which are local definitions and global definitions.

- *Local definitions*: Communities are parts of the graph with a few ties with the rest of the system. To some extent, they can be considered as separate entities with their own autonomy. So, it makes sense to evaluate them independently of the graph as a whole. Local definitions focus on the subgraph under study, including possibly its immediate neighborhood, but neglecting the rest of the graph. Four types of criterion are used in local definitions:

complete mutuality, reachability, vertex adjacency and *comparison of internal versus external cohesion.* Complete mutuality is a very strict condition, as it requires each vertex of a community to be adjacent to the rest of the vertices in the community, which translates the definition of community to a clique. A clique with a missing link should certainly be considered as a very cohesive group and therefore a community, but it would not be considered a community under this recipe. Reachability offers a more relaxed way to express the latter condition, as it requires the vertices to have a path between them that does not exceed a certain number of steps to consider them a community. Vertex adjacency is another criterion for subgraph cohesion. The idea is that a vertex must be adjacent to some minimum number of other vertices in the subgraph. Finally, the comparison of internal versus external cohesion is usually done by a fitness measure, which expresses to which extent a subgraph satisfies a given property related to its cohesion. The larger the fitness, the more defined the community is. The simplest fitness measure for a cluster is its intra-cluster density $\rho^{int}(\mathcal{C})$.

- *Global definitions*: Communities can also be defined with respect to the graph as a whole. This is reasonable in those cases in which clusters are essential parts of the graph, which cannot be taken apart without seriously affecting the functioning of the system. The literature offers many global criteria to identify communities. In most cases they are indirect definitions, in which some global property of the graph is used in an algorithm that delivers communities at the end. However, there is a class of proper definitions, based on the idea that a graph has community structure if it is different from a random graph. A random graph à la Erdös–Rényi (where links between nodes appear with constant probability) is not expected to have community structure, as any two vertices have the same probability to be adjacent, so there should be no preferential linking involving special groups of vertices. Therefore, one can define a *null model*, i.e. a graph which matches the original in some of its structural features, but which is otherwise a random graph. The null model is used as a term of comparison, to verify whether the graph at study displays community structure or not. The most popular null model is that proposed by Newman and Girvan and consists of a randomized version of the original graph, where edges are rewired at random, under the constraint that the expected degree of each vertex matches the degree of the vertex in the original graph [35]. This null model is the basic concept behind the definition of *modularity*, a quality function which evaluates the goodness of partitions of a network into communities, which we will see in the following sections.

2.2 Community Detection Methods

The aim of community detection methods is to find partitions of nodes in a network. A partition is a division of the network in clusters, such that each vertex belongs to one cluster. As we can observe from real networks, some vertices could be shared among different communities. A division of a graph into overlapping communities is called a cover. In some cases, communities have hierarchical structure, this happens when the graph has different levels of organization. Then the communities are divided into new smaller communities and so on. And last but not least, it is also possible that communities are different levels of organization are part hierarchical, part overlapped. In this last case multiresolution methods are necessary. One should really analyze the system one has in hand in order to choose, from all the community detection methods available, the one that suits better the problem one wants to solve. Next, the most popular classification of community detection methods is presented.

2.2.1 Classical Methods

Graph partitioning

The problem of graph partitioning consists in dividing the vertices in g groups of predefined size, such that the number of edges lying between the groups is minimal. The number of edges running between clusters is called the *cut size*. Specifying the number of clusters of the partition is necessary. If one simply imposed a partition with the minimal cut size, and left the number of clusters free, the solution would be trivial, corresponding to all vertices ending up in the same cluster, as this would yield a vanishing cut size. Specifying the size is also necessary, as otherwise the most likely solution of the problem would consist of separating the lowest degree vertex from the rest of the graph, which is quite uninteresting. Graph partitioning is a fundamental issue in parallel computing, circuit partitioning and layout, and in the design of many serial algorithms, including techniques to solve partial differential equations and sparse linear systems of equations. The Kernighan–Lin algorithm [28] is one of the earliest methods proposed and is still frequently used, often in combination with other techniques. The authors were motivated by the problem of partitioning electronic circuits onto boards: the nodes contained in different boards need to be linked to each other with the least number of connections. It works by swapping nodes from cluster to cluster while minimizing the cut size.

In general, very little is known about the community structure of a graph. It is uncommon to know the number of clusters in which the graph is split, or other indications about the membership of the vertices. In such cases clustering procedures like graph partitioning methods can hardly be of help, and one is forced to make some reasonable assumptions about the number and size of the clusters, which are often unjustified.

Hierarchical clustering

The network may have a hierarchical structure, i.e. may display several levels of grouping of the vertices, with small clusters included within large clusters, which are in turn included in larger clusters, and so on. Social networks, for instance, often have a hierarchical structure. In such cases, one may use hierarchical clustering algorithms, that is clustering techniques that reveal the multilevel structure of the graph. It is essential to define a measure of similarity between vertices, and calculate this similarity between all pairs of vertices, even if they are not connected by a link. According to the way the similarity is used, hierarchical methods can be classified into two categories. In *agglomerative algorithms*, one starts from the vertices as separate clusters and creates the communities by putting together those nodes that are similar enough. The other type of algorithms follow the opposite approach, as *divisive algorithms* start from the graph as a whole cluster and the division is done by removing the edges between nodes with low similarity. The drawback of this method is that the assumption of hierarchical structure could not be exact.

Partitional Clustering

Partitional clustering is another traditional class of methods for classifying data points into clusters. The number of clusters has to be decided beforehand, say k clusters. The data points are embedded into a metric space, therefore a distance function between pairs of points can be defined in order to calculate the dissimilarity between these points or vertices. The goal is to separate the points into k vertices in a way that minimizes the distance function chosen. One of the most used function is the *minimum k-clustering*, in which the cost function is the diameter of the clustering. The diameter is the distance between the two farthest points in a cluster, so the aim in this methodology is to classify the points in a way that the diameters of each cluster are minimum. Other distance functions widely used are *k-clustering sum, k-center* or *k-median.* Again, the guess about the number of partitions determines the output, and this is something difficult to assess *a priori.*

Spectral Clustering

The family of spectral clustering includes all methods that perform the clustering of the data points by using the eigenvectors of matrices. The transformation to be done consists in calculating the elements of the eigenvectors of the original set of objects. The result is a set of points in space, which will then be clustered using any other clustering technique. Methods in spectral clustering are able to separate data points that could not be resolved by directly applying other traditional methods, as the change of representation induced by the eigenvectors could make the cluster properties of the initial data set much more evident.

2.3 Modularity

In the first section of this chapter we mentioned modularity as a quality function. It is used to assess the user on the quality of the partitions obtained through a community detection algorithm. By assumption, high values of modularity indicate good partitions, so the partition which has maximum value of modularity in a graph should be considered the best one, or at least a very good one. This is the reasoning behind modularity optimization, by far the most used technique for community detection. Note that modularity does not assume any *a priori* information about the number of partitions, the number of partitions is an output of the optimization problem. It is known as modularity optimization instead of modularity maximization because it is impossible to find the highest value of modularity in a reasonable amount of time, since the number of different partitions is equal to the Bell [5] or exponential numbers, which grow at least exponentially in the number of nodes N.

The way in which modularity evaluates the goodness of a partition is by comparing its communities with a null case. This null case is a representation of the same network we are willing to detect communities in, but with some variations. The modification made in the original network to form the null model is that it has been randomly rewired preserving the degree of each node. That means changing the links within the network but keeping to each node its original degree. Note that for weighted networks the strength (sum of weights per node) has to be preserved. This null model provides us with a network very similar to the original one but as it is random, it is expected not to have communities. The value of modularity represents the deviation of the original network from the randomized version for the same partitioning. If the network does not have community structure, it will be very similar to a random one and the value of modularity will be low. If the network has well defined communities, then the deviation from the null case is more accentuated and the value of modularity is high.

Given a network partitioned into communities, being C_i the community to which node i is assigned, the mathematical definition of modularity is expressed in terms of the weighted adjacency matrix w_{ij}, that represents the value of the weight in the link between nodes i and j, this weight would be 0 if no link existed, and the strengths $w_i = \sum_j w_{ij}$ as [32]

$$Q = \frac{1}{2w} \sum_i \sum_j \left(w_{ij} - \frac{w_i w_j}{2w} \right) \delta(C_i, C_j)\,, \tag{2.3}$$

where the Kronecker delta function $\delta(C_i, C_j)$ takes the values, 1 if node i and j are into the same community, 0 otherwise, and the total strength $2w = \sum_i w_i$. The modularity of a given partition is then, the probability of having edges falling within groups in the network minus the expected probability in the equivalent (null case) network with the same number of nodes, and edges placed at random preserving the nodes' strength.

The generalization of Eq. (2.3) to directed networks simply reads [1]

$$Q = \frac{1}{2w} \sum_i \sum_j \left(w_{ij} - \frac{w_i^{\mathrm{out}} w_j^{\mathrm{in}}}{2w} \right) \delta(C_i, C_j)\,, \tag{2.4}$$

where $w_i^{\mathrm{out}} = \sum_k w_{ik}$ is the output strength of node i and $w_j^{\mathrm{in}} = \sum_k w_{kj}$ is the input strength of node j.

As we have seen in previous sections, real world networks have often weighted links, and possibly positive and negative values coexist. This leads us to have to adapt the formulation of modularity for the case of weighted undirected signed networks, leaving the directed case to the end of this section. To do so, we have to introduce the following notation: first let us separate the positive and negative weights:

$$w_{ij} = w_{ij}^{+} - w_{ij}^{-}\,, \tag{2.5}$$

where

$$w_{ij}^{+} = \max\{0, w_{ij}\}\,, \tag{2.6}$$

$$w_{ij}^{-} = \max\{0, -w_{ij}\}\,. \tag{2.7}$$

The positive and negative strengths are given by

$$w_i^{+} = \sum_j w_{ij}^{+}\,, \tag{2.8}$$

$$w_i^{-} = \sum_j w_{ij}^{-}\,, \tag{2.9}$$

and the positive and negative total strengths by

$$2w^+ = \sum_i w_i^+ = \sum_i \sum_j w_{ij}^+ \,, \tag{2.10}$$

$$2w^- = \sum_i w_i^- = \sum_i \sum_j w_{ij}^- \,. \tag{2.11}$$

Obviously it is satisfied that

$$w_i = w_i^+ - w_i^- \tag{2.12}$$

and

$$2w = 2w^+ - 2w^- \,. \tag{2.13}$$

With these definitions at hand, the connection probabilities with positive and negative weights are respectively

$$p_i^+ = \frac{w_i^+}{2w^+} \,, \tag{2.14}$$

$$p_i^- = \frac{w_i^-}{2w^-} \,. \tag{2.15}$$

Now there are two terms which contribute to modularity: the first one takes into account the deviation of actual positive weights against a null case random network given by probabilities p_i^+, and the other is its counterpart for negative weights. Thus, it is useful to define

$$Q^+ = \frac{1}{2w^+} \sum_i \sum_j \left(w_{ij}^+ - \frac{w_i^+ w_j^+}{2w^+} \right) \delta(C_i, C_j) \,, \tag{2.16}$$

$$Q^- = \frac{1}{2w^-} \sum_i \sum_j \left(w_{ij}^- - \frac{w_i^- w_j^-}{2w^-} \right) \delta(C_i, C_j) \,. \tag{2.17}$$

The total modularity must be a trade off between the tendency of positive weights to form communities and that of negative weights to destroy them. If we want that Q^+ and Q^- contribute to modularity proportionally to their respective positive and negative strengths, the final expression for modularity Q is

$$Q = \frac{2w^+}{2w^+ + 2w^-} Q^+ - \frac{2w^-}{2w^+ + 2w^-} Q^- \,. \tag{2.18}$$

An alternative equivalent form for modularity Q shown in [20] is

$$Q = \frac{1}{2w^+ + 2w^-} \sum_i \sum_j \left[w_{ij} - \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{w_i^- w_j^-}{2w^-} \right) \right] \delta(C_i, C_j) \,. \tag{2.19}$$

The main properties of Eq. (2.19) are: without negative weights, the standard modularity is recovered; modularity is zero when all nodes are together in one community; and it is antisymmetric in the weights, i.e. $Q(C, \{w_{ij}\}) = -Q(C, \{-w_{ij}\})$.

The extension of modularity to signed directed networks is simply obtained by the substitutions

$$w_i^{\pm} \quad \to \quad w_i^{\pm,\mathrm{out}} = \sum_k w_{ik} \,, \tag{2.20}$$

$$w_j^{\pm} \quad \to \quad w_j^{\pm,\mathrm{in}} = \sum_k w_{kj} \,. \tag{2.21}$$

This form of modularity allows to deal with weighted signed and even directed networks, and has been widely used in a lot of algorithms that optimize modularity.

2.3.1 Optimization algorithms for modularity

Here there is a summary of the main algorithms used to this end:

- *Greedy techniques.* The first algorithm devised to optimize modularity was a greedy method by Newman [33]. Just as in the hierarchical agglomerative methods explained earlier, this algorithm forms communities by merging groups of nodes starting out from all nodes being isolated. Instead of doing that according to some measure of similarity of the nodes, now the groups of vertices are merged such that modularity increases after the merging. Other algorithms that use greedy procedures to optimize modularity are [6, 41, 18], all of them perfom a local search in the space of solutions taking the best available solution at each step.

- *Simulated annealing.* It is a stochastic greedy method in which the search on the space of solutions is not always optimal at each step. The algorithm is controlled by a parameter β, called *temperature*, that determines the probability of choosing a solution in the space of partitions that does not improve the last solution found. This temperature is going to zero as the algorithm evolves making the process deterministic at the end. The advantage in front of greedy algorithms relies on the relative independence of the initial state on the final outcome. Simulated annealing can be used for small graphs, with up to about 10^4 vertices.

- *Extremal optimization.* As an alternative for simulated annealing, Boettcher and Percus proposed an heuristic that would have an accuracy comparable to simulated annealing but in substantially less computer time [8]. This technique was first used to optimize modularity by Duch and Arenas [12] and its basis is to calculate locally the contribution of each vertex to the global modularity. The starting point is a random partition of the graph into two groups with the same number of vertices. Then, at each time step, the vertex with lower contribution to the global modularity is shifted to the other community. As the partition now has changed, the local values of modularity have to be recalculated. This two steps are repeated until there is no further improvement of modularity.

- *Spectral optimization.* Modularity can be optimized by calculating the eigenvectors and eigenvalues of a modularity matrix B, whose elements are $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$. One has to look for the eigenvector of B with the highest positive eigenvalue, and group the vertices according to the signs of its components.

- *Tabu optimization.* This method to optimize the modularity is based on Tabu search [3, 19].The algorithm proceeds as follows: the starting point is a random partition of the network, and we have an operator called *move* that allows us to transform a certain partition into a new one by moving randomly a node from one community to another. With the starting partition and the *move* operator we can calculate, from the initial partition, a set of new partitions that we will call its neighbors. Then, we choose, from all the neighbor partitions, the one that has higher modularity, and discard the initial one. The neighborhood from this new current partition is calculated again and a new partition is chosen until there is no further improvement of modularity. However, during this process, we could get trapped in local optima or even in cycles. In order to avoid that, a list of tabu moves is used. This tabu list stores and forbids the most recently accepted moves and it is updated as the algorithm proceeds, and it also controls that the node that has been moved lastly and has lead to an improvement in modularity is not moved back again in the next step [3].

For the purpose of the work presented in this document, I have concentrated mainly on two methods explained above: the extremal and the tabu optimization. Moreover, they have been used in combination with the fast optimization greedy technique, in order to achieve better modularity results.

2.4 Limitations of modularity

A lot of work has been done to devise reliable techniques to maximize modularity [11, 12, 21, 33, 34, 38]. However, very little has been done to analyze the concept of modularity itself and its reliability as a method for community detec-

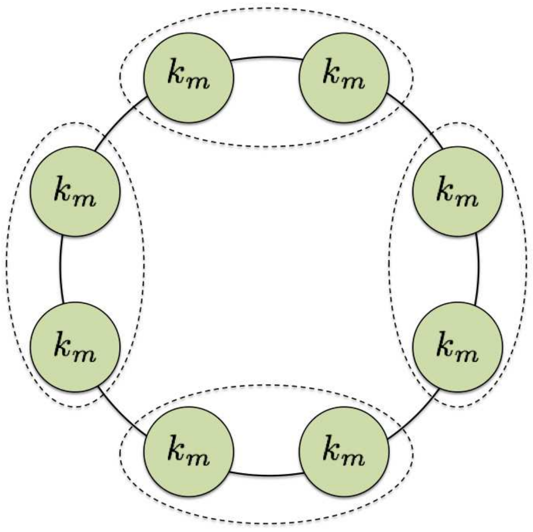

Figure 2.3: Network formed by n cliques of m nodes each, joint by a single link. For $n = 30$ and $m = 5$ the modularity optimization solution groups the cliques in pairs.

tion. To a large extent, the success of modularity as a quality function to analyze the modular structure of complex networks, relies on its intrinsic simplicity. The researcher interested in this analysis is endowed with a non-parametric function to be optimized: modularity. The result of the analysis will provide a partition of the network into communities such that the number of edges within each community is larger than the number of edges one would expect to find by random chance. As a consequence, each community is a subset of nodes more connected between them than with the rest of the nodes in the network. However, recently it has been shown that modularity is not the panacea of the community detection problem; in particular it suffers from a resolution limit that avoids grasping the modular structure of networks at some scales of resolution. As Fortunato and Barthélemy pointed out, there are some situations in which modularity cannot detect certain communities although they belong to networks with clear community structure [16].

See for example the network in Fig. 2.3, in which each circle represents a clique formed by m nodes and all cliques are connected in a ring, by a single link. The traditional modularity optimization result for this setup where n=30 and m=5 would be that each community is formed by two cliques, as it is represented by the dotted lines. The authors in [16] pointed out that this was not the right division of the network into communities, as the cliques are the most densely connected structures possible and they should be considered a community alone. The authors were right about this counter-intuitive behavior of modularity; however some clarification should be made. Modularity involves an implicit definition of what a community is. In fact, it is considered that a community is the result of maximizing modularity, with no further assumptions of what a community should look like. From this point of view then we should say that there exist some substructures of networks that although they might be considered communities according to other definitions of the term, they does not suit the modularity definition. The reason for modularity not detecting certain substructures of the network properly in this setup can be explained mathematically. For the setup in Fig. 2.3 each clique is a complete graph K_m which has m nodes and $m(m-1)/2$ links. If we assume that there are n cliques (being n an even number), the network then has a total of $N = nm$ nodes and $L = nm(m-1)/2 + n$ links. Starting out from the original formulation of modularity, the contributions to modularity made by the communities formed by one or two cliques can be calculated and are as follows:

$$Q_{single} = 1 - \frac{2}{m(m-1)+2} - \frac{1}{n} \tag{2.22}$$

$$Q_{pairs} = 1 - \frac{1}{m(m-1)+2} - \frac{2}{n} \tag{2.23}$$

$$Q_{single} > Q_{pairs} \iff m(m-1) + 2 > n \tag{2.24}$$

Equations 2.22 and 2.23 measure the contribution to modularity of the partitions formed by communities of two cliques or one clique respectively. As shown in Eq. 2.24, for modularity detecting the communities as single cliques, the contribution of Q_{single} should be greater than Q_{pairs}, but this depends on a relation between the number of cliques n and the number of nodes each clique has m. In this example, m and n are independent variables, so it is easy to find a case in which the inequality in 2.24 is not satisfied. For example, for the values $m = 5$ and $n = 30$, the modularity values are $Q_{single} = 0.876$ and $Q_{pairs} = 0.888$, being this latter value greater than Q_{single}. Therefore the communities found by a modularity optimization should group the cliques in two. This would also happen for larger values of n, if m is fixed.

Although that finding does not act in favor of modularity being considered a reliable method for community detection, it is worth saying that the setups presented are synthetic networks whose structure does not have much in common with real world networks. Therefore, and taking into account the great results obtained through optimization of modularity in more realistic scenarios, we feel it is worth trying to deal with the resolution limit in modularity, and to try to solve it or at least, palliate its effects.

2.5 Multi-scale Community Detection Methods

Due to the recent findings about the resolution limit of modularity, some scientists have tried to adapt the classical methods for community detection to be able to diminish the effect of this limit of resolution. To do so, one of the most well-known techniques is to try to analyze the community structure from different levels of resolution. In this section we will explain the benefits of this kind of approach as well as the two most famous multi-scale algorithms that are widely used nowadays.

2.5.1 Multiple scales of resolution in complex networks

Facing a famous painting by Salvador Dali (see Fig. 2.4) we can observe that in complex systems there is not only one scale of resolution which may be interesting to analyze, but there are many which coexist at the same time and contain relevant information. If we observe the painting close enough, we can see that the picture is actually formed by small tiles which have drawings in them (see Fig. 2.4 Left). They are the minimum unit of the painting and altogether they form what we should call the *microscale*. Instead, if we place ourselves 20 meters from the painting, as the author suggests, what we see is that all those tiles join to form the face of Abraham Lincoln (see Fig. 2.4 Right). In that position, we would observe the *macroscale* of the system. However, there exist some intermediate scales of resolution between the microscale and the macroscale, and we will call them the *mesoscales*. In one of those intermediate scales we can see what the author is trying to represent, which is his wife Gala, looking at the sea through a window.

Using the mapping of complex systems into complex networks, the three terms introduced before have a precise meaning. The microscale of a network is formed by the features of the individual nodes, while the macroscale is defined by the statistical properties of the whole network. Then, the mesoscales would be the result of analyzing the structure of the network with an intermediate resolution, and they would be represented by the substructure of the network, say the communities or modules. Successfully detecting the mesoscales of complex networks is a relevant issue as these internal substructures play an important role for the understanding of the relationship between topology and functionality of the networks.

There are plenty of community detection methods that work well for detecting communities in networks. However, most of them deliver only a single snapshot of the substructure of the network, that is, a single scale of the mesoscales. What we

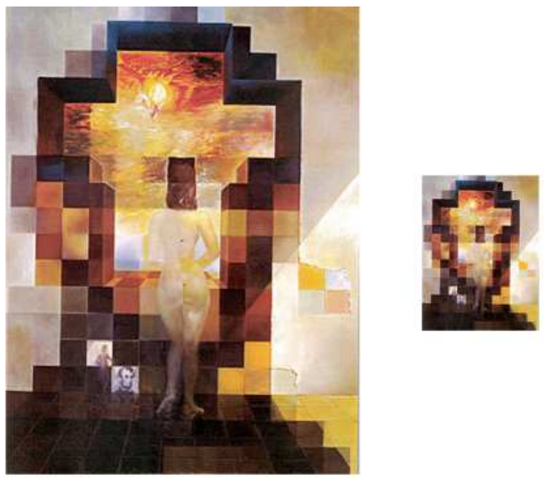

Figure 2.4: *"Gala contemplating the Mediterranean sea which at twenty meters becomes a portrait of Abraham Lincoln"*, by Salvador Dali, 1974. Left, at closer distance, and right, at larger distance.

will try to stress is that one single division of the network into communities (one partition) is not enough to understand completely the structure of the network.

Fig. 2.5 is the representation of a synthetic network made to illustrate the fact that sometimes several mesoscales coexist. It is a network with hierarchical community structure, with 256 nodes. Each node is connected to other nodes in the smallest communities by 13 links, to the nodes within the big communities with 4 links and to a node of the rest of the network with one single link. If we were about to perform community detection on this network, what partition would we find? From a topological point of view we can see that there are two possible solutions to the problem, one that divides the network into four communities of 64 nodes each, and the other that divides the network into 16 communities formed by 16 nodes each (see Fig. 2.6). The question is then, which one of these two solutions is the correct one? The answer is that there is not such thing as a good partition and a bad one, both solutions coexist in the same network and therefore both are representative of its internal structure, it would only depend on the final purpose of the clustering which of the two partitions is more convenient. However, as it is normal not to have any previous knowledge about the communities in a network, it is important to have a criterion to decide which one of the partitions is more representative in terms of structure. This criterion should depend on the community detection method used and will be discussed in the following section.

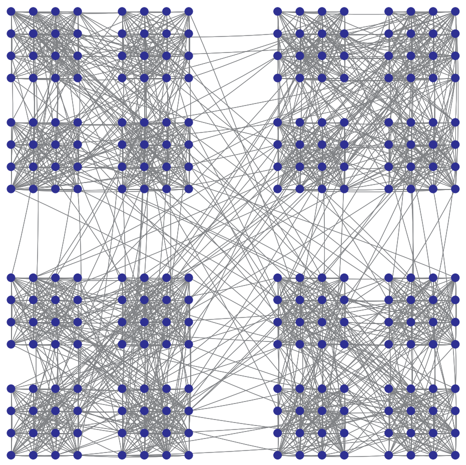

Figure 2.5: Synthetic network that presents internal community structure at different scales, formed by 256 nodes. Each node has 13 links within the smaller communities, 4 links within the big communities, and 1 link connecting this node to a node in any of the three other big communities.

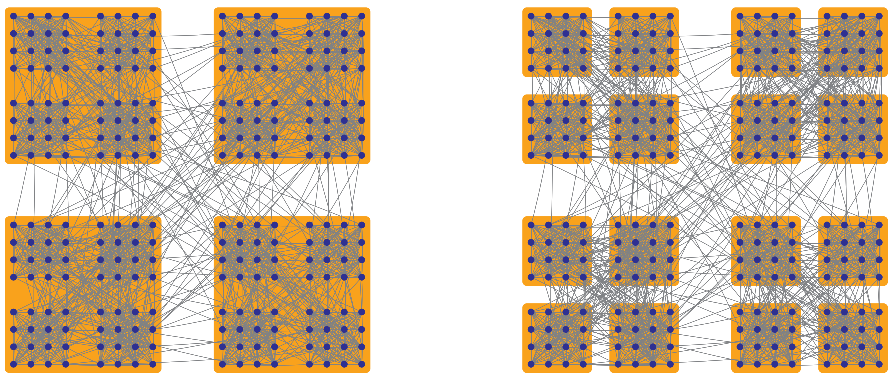

Figure 2.6: Two possible partitions of the same network. Left, the division into four communities of 64 nodes each. Right, the division into 16 communities of 16 nodes each.

2.5.2 Multiresolution algorithms

RB algorithm

There are other multi-scale methods for optimizing modularity. One of them is the one by Reichardt & Bornholdt (RB) [40]. Their approximation starts out from the original formulation of modularity as well, consists in adding a parameter γ to the null case term. This is the parameter that will be tuned and will provide the access to the mesoscale. The formulation is shown in Eq.2.25.

$$Q = \frac{1}{2w} \sum_i \sum_j \left(w_{ij} - \gamma \frac{w_i w_j}{2w} \right) \delta(C_i, C_j) \,, \tag{2.25}$$

What it is achieved by tuning the γ parameter is to move the change the null term at each step. Consequently, different resolutions are obtained for each γ value, but the semantics of modularity is not preserved. Furthermore, there is a problem when using negative values of γ. This factor is affecting the null term, which is the probability of two nodes having a link between then in a randomized version of the network. When the γ goes negative, the probability becomes negative and loses its meaning. This method and the AFG algorithm presented previously are not equivalent, so there is not a translation between γ and r. For a more detailed comparative, see [3].

Original AFG algorithm

One of the most well-known solutions for performing modularity optimization at different levels of resolution is the Arenas-Fernández-Gómez (AFG) algorithm [3]. This is the method we will refer to in the following sections. This method is based on the idea that, to analyze the network at different levels of resolution we need to be able to observe the network from different *distances*. Imagine you have got to take a picture from an elephant next to a mouse. With one single snapshot it is impossible to capture all the details of both animals, as they are very different in size. On the contrary, you would have to make two different pictures and play with the zoom to achieve the information you want. How to achieve that magnifying lens effect when optimizing modularity on a network? The idea is to introduce a global magnitude to the network, but in a way that the statistical properties of the network (degree distribution, degree-degree correlation, spectrum...) are preserved. This global magnitude will then be a self-loop added to each node, with a value of r equal to all nodes in the network. In that way, the modification to the network remains in the diagonal of the (weighted) adjacency matrix and does not change any of the statistical properties. The formulation for modularity containing the parameter r is explained below. First of all, we have to rewrite (2.3) in terms of contribution of modules instead of nodes:

$$Q = \sum_{s=1}^{m} \left(\frac{w_{ss}}{w} - \left(\frac{w_s}{2w} \right)^2 \right) , \tag{2.26}$$

where the sum is over the m modules of the partition, w_{ss} is the internal strength of module s and w_s the total strength of module s. For unweighted networks w_{ss} reduces to the number of internal links and w_s to the sum of degrees of the nodes in module s.

The problem now is how to increase the strength of nodes without altering the topological characteristics of the original network. We solve this problem by rescaling the topology defining $\mathbf{W}_r$, from the original weighted adjacency matrix $\mathbf{W}$ of the graph with entries w_{ij}, as follows

$$\mathbf{W}_r = \mathbf{W} + r\mathbf{I} , \tag{2.27}$$

where $\mathbf{I}$ is the identity matrix. In terms of graphs, this new matrix represents the original network with self-loops of weight r for every node. Note that the prescription in (2.27) supposes a constant shift (translation) r of the strength of each node.

Denoting Q_r the modularity of the network at scale r, the equivalent expression to (2.26) reads

$$Q_r = \sum_{s=1}^{m} \left(\frac{2w_{ss} + n_s r}{2w + Nr} - \left(\frac{w_s + n_s r}{2w + Nr} \right)^2 \right) . \tag{2.28}$$

The parameter r accounts for *resistance parameter*, which should be interpreted as the resistance a node has to form part of a community. Loosely speaking, when the resistance parameter is positive and has its maximum value (r_{max}), the nodes are reinforced and each would form its own community. This means that the partition obtained is dividing the network in singletons. On the other case, if the value of r is negative and has its minimum value (r_{min}), the nodes are *weak* and they have to attach to other nodes in the network in order to form a community, therefore the partition obtained is formed by a single community containing all nodes. Tuning this parameter ranging from r_{min} to r_{max} and performing a modularity optimization at each step, we are provided with a partition for each level of resolution of the network and we are able to observe the whole mesoscale as intended. Note that the original formulation of modularity is recovered when $r = 0$.

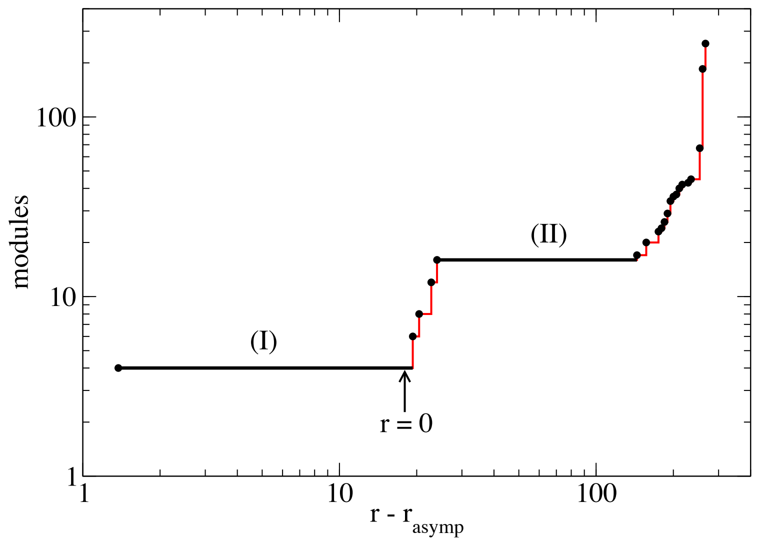

Figure 2.7: Representation of the number of communities of the network as a function of the resistance parameter. The length of the plateaus is representative of the relevance of each partition.

The topological scale corresponding to all nodes separated is found by maximizing $Q_{r_{\max}}$, where $r_{\max}$ is the smallest positive value of r that satisfies

$$w_{ij} < \frac{(w_i + r)(w_j + r)}{2w + Nr}, \quad \forall i \neq j \tag{2.29}$$

The topological scale corresponding to a unique module formed by the whole network is found by maximizing $Q_{r_{\min}}$, where $r_{\min}$ has a lower bound defined by the asymptote $r_{\text{asymp}} = -\frac{2w}{N}$.

To illustrate the performance of this algorithm, we will use the synthetic network in Fig. 2.5. As we have previously pointed out, there are two possible configurations of communities when observing the mesoscale of that network. So a good multi-scale algorithm should be able to reveal both solutions. Indeed, what it is found by using the AFG algorithm is a partition containing 4 communities with 64 nodes each and a partition formed by 16 communities of 16 nodes each, among other partitions that divide the nodes in different setups.

Plotting the different partitions as a function of r is providing us with the criterion needed to distinguish which of the partitions obtained are more or less relevant. If we observe Fig. 2.7 we realize that there are two partitions that remain unchanged during more values of r than others. The first plateau (I) corresponds to the division into four communities, and the second (II) is the division of the network in 16 communities. We can observe that both partitions are more stable than the rest of possible divisions of the network, which is giving us information about the relevance of those two configurations.

2.5.3 Evolution of AFG method for weighted signed networks

The first contribution of this Master Thesis to the object of study considered, consisted in to provide a new formulation of the AFG method to deal with weighted and signed networks. This aspect is crucial to have a versatile algorithm for resolving the multiple scales of the community structure of complex networks in the most general scenario. This modification is specially interesting to analyze correlation data sets as complex networks given that these kind of data is intrinsically signed (correlated and anti-correlated values).

Let us start by resuming the generalization of modularity Eq. (2.19) for undirected weighted signed networks:

$$Q = \frac{1}{2w^+ + 2w^-} \sum_i \sum_j \left[w_{ij} - \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{w_i^- w_j^-}{2w^-} \right) \right] \times \delta(C_i, C_j) \,. \tag{2.30}$$

where

$$w_i^+ = \sum_{j, w_{ij}>0} w_{ij} \,, \tag{2.31}$$

$$w_i^- = \sum_{j, w_{ij}<0} |w_{ij}| \,, \tag{2.32}$$

are the positive and negative strengths of node i, and

$$2w^+ = \sum_i w_i^+ \,, \tag{2.33}$$

$$2w^- = \sum_i w_i^- \,, \tag{2.34}$$

the positive and negative total strengths respectively. Please note that these four strengths are defined to be non-negative.

To simplify the notation, we make use of the modularity matrix

$$B_{ij} = w_{ij} - \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{w_i^- w_j^-}{2w^-} \right) \,, \tag{2.35}$$

therefore

$$Q = \frac{1}{2w^+ + 2w^-} \sum_{i=1}^{N} \sum_{j=1}^{N} B_{ij} \delta(C_i, C_j) \,. \tag{2.36}$$

Following [3], the analysis of the mesoscale is performed with the addition of a common self-loop to all the nodes in the network. The boundaries of the mesoscale are the *macroscale*, a partition in which all nodes belong to the same community, and the *microscale*, a partition in which each node is isolated in its own community. The determination of these boundaries is equivalent to finding two values of the self-loops, $r_{\min}$ and $r_{\max}$, for which the maximum of modularity $Q_{\mathrm{AFG}}(r)$ is achieved at the macroscale and microscale respectively. The solution is quite simple: if all the non-diagonal terms of the modularity matrix are positive or zero, modularity is optimized at the macroscale, and if they are negative, it is optimized at the microscale. Diagonal terms are irrelevant since $\delta(C_i, C_i) = 1$ for all nodes.

If we introduce a positive self-loop r^+, the modularity matrix becomes

$$B_{ij}^{\mathrm{AFG}}(r^+) = w_{ij} + r^+\delta_{ij} - \left(\frac{(w_i^+ + r^+)(w_j^+ + r^+)}{2w^+ + Nr^+} - \frac{w_i^- w_j^-}{2w^-} \right) , \tag{2.37}$$

and with a negative self-loop $-r^-$

$$B_{ij}^{\mathrm{AFG}}(-r^-) = w_{ij} - r^-\delta_{ij} - \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{(w_i^- + r^-)(w_j^- + r^-)}{2w^- + Nr^-} \right) . \tag{2.38}$$

The existence of $r_{\max}$ is straightforward, since $B_{ij}^{\mathrm{AFG}}(r^+) \sim -r^+ < 0$ for large enough r^+ and $i \neq j$. Its determination is just an exercise of solving the system of inequations $B_{ij}^{\mathrm{AFG}}(r^+) \leq 0$ for $i < j$, and taking the smallest solution as $r_{\max}$. More precisely,

$$r_{\max} = \max_{\substack{i<j \\ D_{ij}^2 \geq 4E_{ij}}} \left(-\frac{D_{ij}}{2} + \frac{1}{2}\sqrt{D_{ij}^2 - 4E_{ij}} \right) , \tag{2.39}$$

where

$$D_{ij} = w_i^+ + w_j^+ - N\left(w_{ij} + \frac{w_i^- w_j^-}{2w^-} \right) , \tag{2.40}$$

$$E_{ij} = w_i^+ w_j^+ - 2w^+\left(w_{ij} + \frac{w_i^- w_j^-}{2w^-} \right) . \tag{2.41}$$

In the same way, $B_{ij}^{\mathrm{AFG}}(-r^-) \sim r^- > 0$ proves the existence of $r_{\min}$, and it is calculated by solving $B_{ij}^{\mathrm{AFG}}(-r^-) \geq 0$ for $i < j$, and taking the largest solution as $r_{\min}$, i.e.

$$r_{\min} = -\max_{\substack{i<j \\ D_{ij}^2 \geq 4E_{ij}}} \left(-\frac{D_{ij}}{2} + \frac{1}{2}\sqrt{D_{ij}^2 - 4E_{ij}} \right) , \tag{2.42}$$

where

$$D_{ij} = w_i^- + w_j^- + N\left(w_{ij} - \frac{w_i^+ w_j^+}{2w^+}\right), \quad (2.43)$$

$$E_{ij} = w_i^- w_j^- + 2w^-\left(w_{ij} - \frac{w_i^+ w_j^+}{2w^+}\right). \quad (2.44)$$

When the network is directed, the analysis of the AFG mesoscale is exactly the same, but with the substitutions

$$w_i^\pm \to w_i^{\pm,\text{out}} = \sum_{k,\pm w_{ik}>0} |w_{ik}|, \quad (2.45)$$

$$w_j^\pm \to w_j^{\pm,\text{in}} = \sum_{k,\pm w_{kj}>0} |w_{kj}|, \quad (2.46)$$

$$D_{ij} \to \frac{1}{2}(D_{ij} + D_{ji}), \quad (2.47)$$

$$E_{ij} \to \frac{1}{2}(E_{ij} + E_{ji}). \quad (2.48)$$

With this analytical development we are able to apply the multi resolution analysis to any type of complex network.

Chapter 3

Data Clustering Using a Multiresolution Community Detection Method

In this chapter we present one of the original works developed. The purpose is to adapt a community detection method that is specially designed to be applied on a network to the problem of data clustering. Data clustering and community detection share the same nature as their final purpose is to classify elements into groups, but there are some differences that make the mapping not so straightforward. In data clustering the raw material is a set of data or elements, each of them usually represented by a multidimensional vector, being each component of the vector one of the features of an element. Each element is isolated in the sense that there is not an explicit relationship between each element and the others, something strictly necessary when building a network, since this is what defines the meaning of a link between two nodes. Furthermore, the higher the dimensionality of the dataset, the more difficult it is for data clustering algorithms to classify them properly, not only because it is computationally more costly but also because the presence of correlated data can mask the correct classification of the elements. In this chapter we will present one of the most famous real datasets that has nowadays become a classical benchmark for evaluating data clustering algorithms, the Iris dataset. Next, we will present the adaption of the AFG algorithm to perform the clustering of this data along with some encouraging results.

Figure 3.1: Pictures of the three subtypes of Iris flowers present in the dataset. From left to right: Iris setosa, Iris versicolor and Iris virginica.

3.1 A Case Study: The Iris Dataset

The Iris flower dataset is a multivariate data set introduced by Sir Ronald Fisher in 1936 as an example of discriminant analysis. Sometimes it is called the Anderson's dataset, as the information was originally collected by Edgar Anderson, an american botanist. The dataset is formed by 150 samples of Iris flowers, which are known to be divided into three groups of 50 each, corresponding to three subtypes of Iris: Iris setosa, Iris versicolor and Iris virginica, see Fig. 3.1 for a picture. For each sample four features were measured, which were the length and width of the sepals and the petals, in centimetres. This dataset was used by Fisher to test the accuracy of a linear discriminant model to classify the three types of flowers.

The reason why this dataset has become a classical benchmark for data clustering algorithms is because the setosa group is linearly separable from the other two, but versicolor and virginica are quite mixed. Even nowadays there is no algorithm that solves this problem correctly in an unsupervised way. In Fig. 3.2 there is a plot of the 150 patterns when taking into account only two of the features at a time. It is clear that in none of the configurations the versicolor and virginica subgroups are clearly distinguishable.

The question that arises is whether a community detection algorithm could be suitable for the solution of this classification problem. Using the mapping from the dataset's multidimensional points to complex networks, it seems viable to use a community detection method. Moreover, a multi-scale community detection algorithm would be specially suitable for this problem, as its classification is not straightforward and it seems that the exploration of the whole mesoscale is necessary in this context. In the following section we will explain the methodology used and the results obtained.

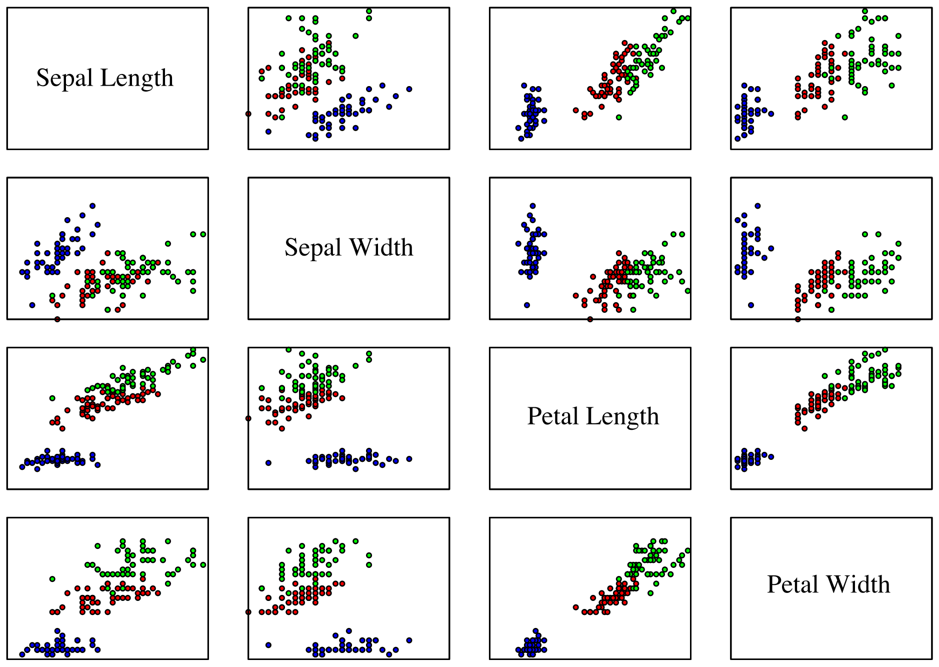

Figure 3.2: Plot of the 150 samples of flowers taking only into consideration two features at a time. Blue dots represent Iris setosa, green correspond to Iris virginica and red is for Iris versicolor.

3.2 AFG Solution to the Iris dataset clustering problem

For the process of extracting the clusters from the Iris dataset we will perform some steps that belong to the data clustering procedure that were explained in section 1.2. The basic steps for data clustering are: choosing the appropriate representation of the data (with or without performing feature selection or extraction), calculating the similarity matrix and applying the clustering algorithm.

3.2.1 Data representation

For the representation of the data, we start out from the original Iris dataset, consisting of 150 patterns of four features each. As seen in the Fig. 3.2 we can discern some pairs of features that seem to facilitate the analysis. For example, when plotting the petal length vs. the petal width, we can see that the two non-linearly separable groups are more easily distinguishable from each other. A first approach then could be to take into consideration only these two features and discard the other two for the analysis from now on. However, there is a more

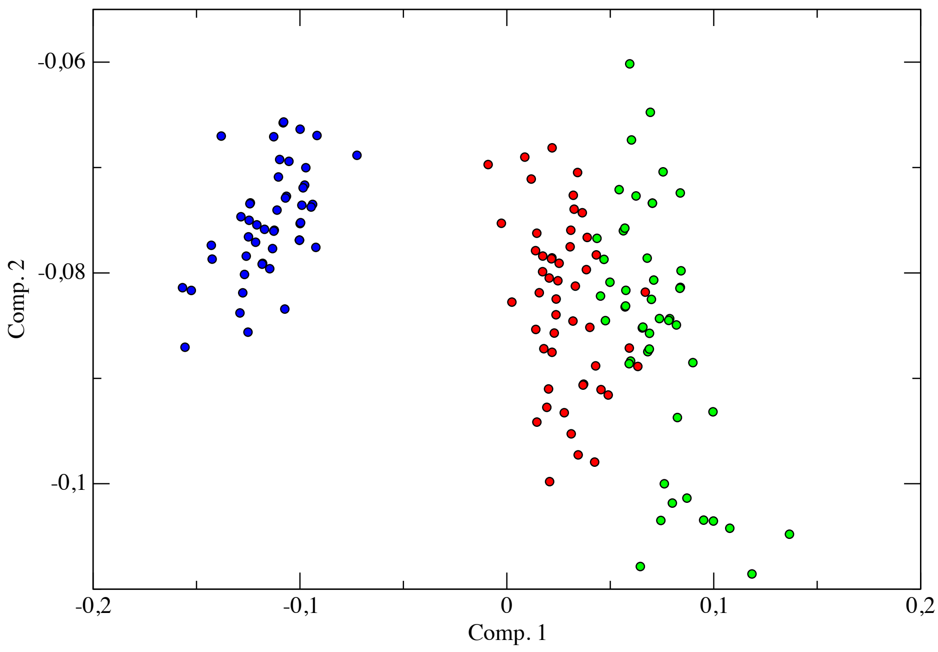

Figure 3.3: Results of the two principal components of the PCA analysis of Iris dataset. We observe a better differentiation between data in this spectral plot. The color code is blue for setosa, green for virginica and red for versicolor.

convenient approach, which consists in performing feature extraction instead of feature selection.

As the information of the sepal length and width is also relevant and maybe could be helpful, what we did is to perform a Principal Component Analysis (PCA) of the four features. PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), the second principal component has the second largest possible variance, and so on. For the analysis of the Iris dataset we choose to work with the first and second principal components resulting from the PCA, see Fig. 3.3.

3.2.2 Similarity matrix as a network

After the transformation of the data, our new dataset is formed by 150 patterns with two features each. Our proposal relies on defining a network, linking the data in such a way that more similar patterns have a stronger link intensity (weight). To calculate the similarity between all pairs of patterns we used the Minkowski distance function (which actually accounts for the dissimilarity between patterns if the distance is large). The Minkowski distance between two points is defined as:

$$\mathrm{dist}(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}, \tag{3.1}$$

where n is the number of dimensions, and x_i and y_i are the i^{th} components of patterns x and y respectively. Note that when $p = 2$ the formula reduces to the Euclidean distance, which is the value of p used in the current work. We tried with some other values of p and have obtained equivalent results, therefore we will use only the Euclidean distance for the purpose of this work.

The idea behind our proposal is to use a community detection algorithm based on modularity. As we have explained before, modularity is a quality function that evaluates the partitions against a null case (the randomized version of the original network). Modularity, a a global function, measures the tendency of a node to form part of a community compared to the forces that could attract the same node to other communities. A way to favor this tendency is to weight the forces in terms of attraction and repulsion, i.e. using signed weights. To facilitate the process of clustering, we want to turn the values of the links into positive and negative values. To do so, we calculate the average of the distances $\bar{d}$, and then the similarity between two nodes is defined as:

$$\mathrm{similarity}(x,y) = \bar{d} - \mathrm{dist}(x,y) \tag{3.2}$$

This matrix of positive and negative values is our similarity matrix. It can be easily represented as a fully connected network, where each node is a pattern and the link between them has a signed value (weight) corresponding to the similarity. With this setup, the AFG multi-scale community detection algorithm can be applied straightforward.

3.2.3 The clustering algorithm

We have used the AFG multi resolution method to find the communities of our network, and identify these communities as the natural clusters on the original

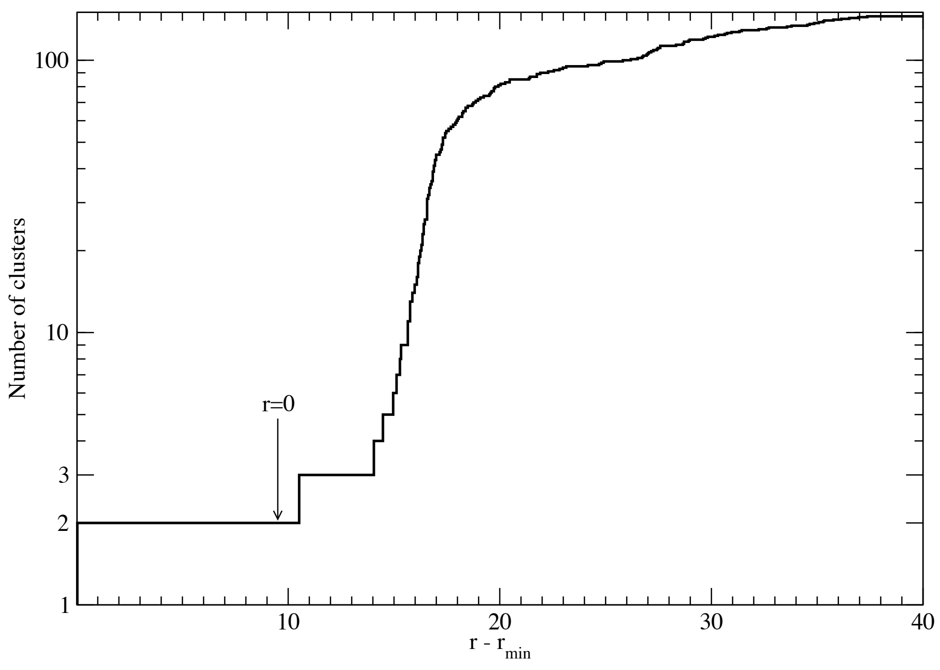

Figure 3.4: Results of the community detection of the Iris dataset. Plot of the number of communities of each partition as a function of the resolution parameter r.

data. The results of the algorithm are presented in Fig. 3.4, and a discussion about its interpretation follows.

Fig. 3.4 is a plot of the partitions obtained (the number of communities) as a function of r, the resistance parameter that tunes the resolution at which the optimization of modularity is performed in the AFG method. The length of the plateaus is giving us an idea of the relevance of each partition obtained. From the plot we can see that the more relevant partition is the one that divides the dataset into two groups, which we know that corresponds to the exact division of 50 patterns of setosa in one community and the rest in another one. The fact that this setup is the most persistant makes total sense as the division of the data into two linearly separable groups is driving the community detection as expected. However, the second most resistant partition corresponds to a division of the dataset into three communities. One of them contains exactly 50 patterns of setosa, and the other two contain virginica and versicolor respectively with some misclassified nodes between them. In this setup, if we compare to the biological classification of the flowers, which is three groups of 50 each, we find that the success ratio is exactly 94.6%. In the plot we can also see that if we set $r = 0$, that is, if we used the original modularity formulation, the result obtained would be a single partition that divides the flowers into the two linearly separable groups, and we would not have more information about further possible divisions of the dataset.

3.2.4 Comparison with other methods

The main objective of the previous sections was to present the methodology to map a problem of unsupervised data clustering into a problem of community detection in complex networks. The performance of the method is reasonably good for the Iris data set classification problem. Although our intention during the development of the current work was not to make a large set of experiments in different data sets, we thought relevant to compare the performance of other algorithms on the same data set to assess the possibilities of the method presented.

To this end we have made an intense research in the literature for different approaches to solve the same problem on the same data set. We have noted that many of the algorithms we find in the artificial intelligence literature use supervised learning to confront the problem. The comparison of results is not possible in the cases that use part of the data for training an intelligent system. Only in a few cases we find unsupervised algorithms to solve the classification of the Iris dataset. We have selected four different algorithms to compare with ours: RB multi resolution method [39], Hierarchical clustering (HC) with complete linkage multidendrogram [15], the k-means algorithm [30] and the proposal by Ng et al. [36].

First we will make the comparison with those algorithms that fix the number of clusters to three groups: k-means and Ng et al. The k-means algorithm simply assumes k spherical clusters in the multi-dimensional data space and looks for their centers using Expectation-Maximization (EM) methods. Although it is unsupervised it needs to be provided with the number of clusters we want to obtain (3 in the case of the Iris dataset). Similarly, in Ng et al. the authors use a k-means to classify the data in spectral form, considering a number of k-eigenvectors, for a specific distance matrix called the "affinity matrix" $A_{ij} = exp(-\|x_i - x_j\|^2/2\sigma^2)$, being x_i the data points. This particularity is considerably less flexible that having the number of clusters undetermined and finding solutions at the multiscale as our algorithm does.

As a second test, we compare our proposal with other two alternatives that in principle can be adapted to the analysis the whole mesoscale, namely the RB and the HC methods. In the RB formulation of mesoscales, a parameter γ is introduced in front of the null-case term to weight its relative importance against the real network, i.e.

$$B_{ij}^{\mathrm{RB}}(\gamma) = w_{ij} - \gamma \left(\frac{w_i^+ w_j^+}{2w^+} - \frac{w_i^- w_j^-}{2w^-} \right) . \qquad (3.3)$$

Method	Errors
k-means	16
Ng et al	14
AFG	8

Table 3.1: Number of missclassified elements using unsupervised methods on the Iris dataset.

It is also possible to have different parameters for the positive and negative null-case terms as in [43], however this leads to a bidimensional analysis of the mesoscales, which is almost unaffordable for most real networks. Thus, we will focus on the single-parameter RB modularity matrix, see Eq. (3.3).

For the HC method, we constructed the hierarchical clustering using complete linkage, where the distance between groups is defined as the distance between the most distant pair of individuals, one from each group. In other words, the distance between two clusters is given by the value of the longest link between the clusters. At each stage of hierarchical clustering, the clusters at minimum distance are merged. Moreover, instead of using the standard pair-group hierarchical clustering approach, we take advantage of a recent development by [15] that allows to solve the non-uniqueness problem when there are tied distances during the agglomeration process. The result, known as a *multidendrogram*, is presented in Fig. 3.5a. We plot the tag number of each specimen of Iris at the leaves of the tree. The analysis of the multidendrogram can be performed as follows: starting from the root of the tree, we can compute the distances between different partitions of the data and analyze each of them separately.

The comparison between the three multi resolution methods AFG, RB and HC can be done by computing the multiple scales of the topology in terms of community structure, screening the values of r in the AFG method, the values of γ in the RB method, and the ultrametric distances in the dendrogram, see Fig. 3.5.

Note that, without negative weights, the macroscale is recovered at $\gamma_{\min} = 0$, and the microscale at the $\gamma_{\max}$ which makes all modularity terms negative. The existence of $\gamma_{\max}$ is guaranteed by the fact that all null-case terms are positive. However, the addition of negative weights makes it possible to have both positive and negative null-case terms, which does not allow to ensure the recovery of macro and microscale. Therefore, RB signed modularity may not cover the whole mesoscale. This is experimentally confirmed in Fig. 3.6 for the Iris data set, where a larger interval of the γ parameter has been analyzed. While Fig. 3.5c only shows

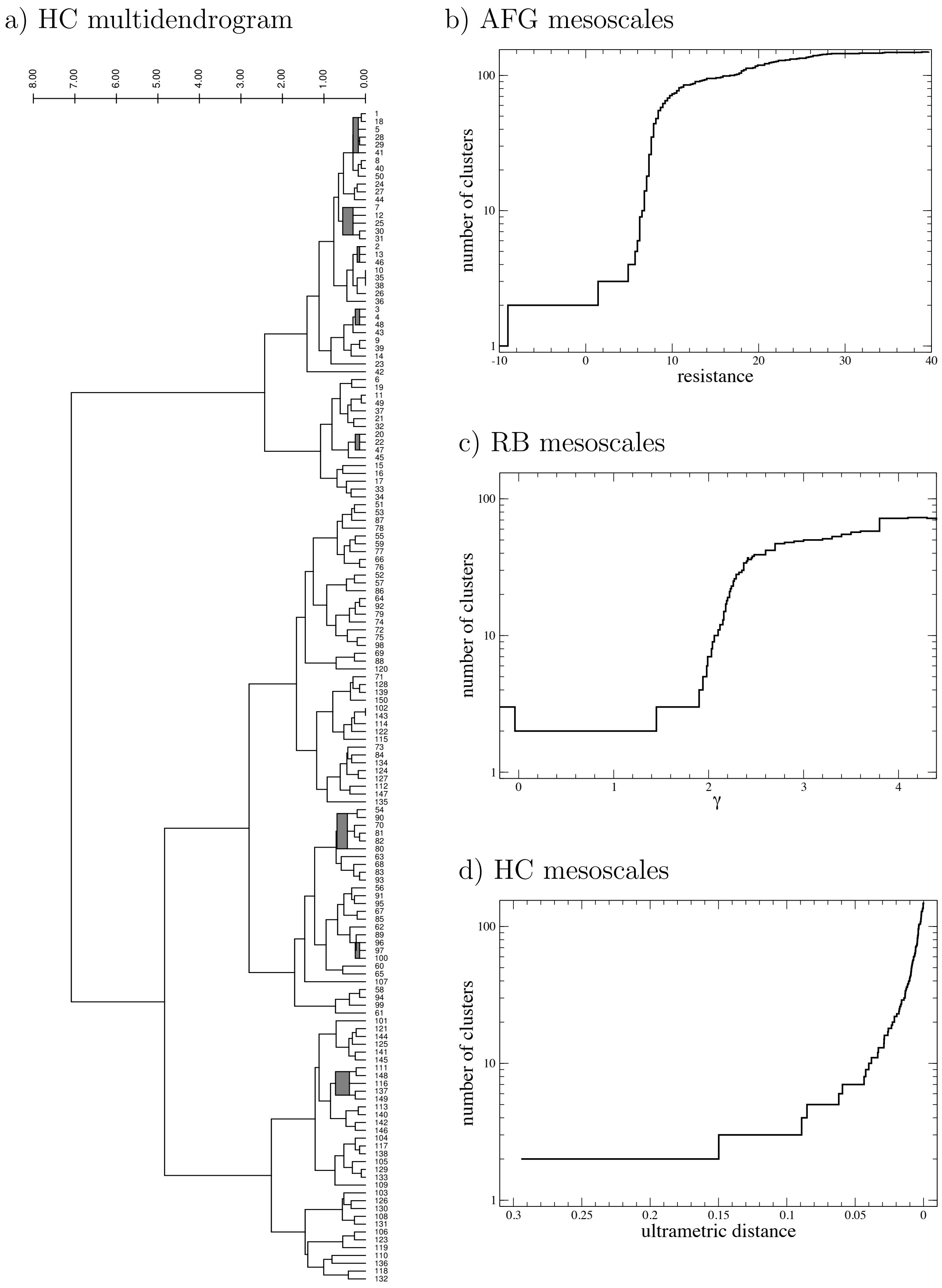

Figure 3.5: Mesoscales of the Iris data set, showing the number of clusters as a function of the resolution parameter, for the three multi resolution schemes: a) HC complete linkage multidendrogram; b) AFG mesoscales; c) RB mesoscales; d) HC mesoscales from the previous multidendrogram.

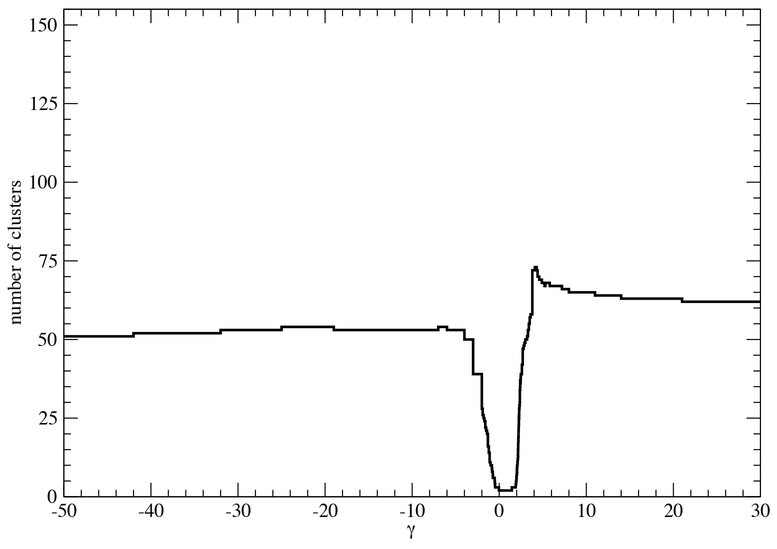

Figure 3.6: Expanded Iris data set RB mesoscales analysis.

the useful part of the mesoscales range, where the number of clusters goes from 2 to 73 ($\gamma \in [0.0, 4.2]$), in Fig. 3.6 we show the inability of RB to find the macroscale (microscale) for lower (larger) values of γ.

Chapter 4

Overcoming the Resolution Limit of Multiresolution AFG Method

Some recent findings point out that the problem of resolution limit in modularity has some other implications besides the problems previously explained. In this chapter we will show the magnitude of this problematic and we will introduce a new algorithm, which is a variation of the AFG method, whose purpose is to overcome the resolution limit or at least, palliate its effects as much as possible.

4.1 Multiresolution community detection and the problem of the resolution limit

In addition to the problem stated in section 2.4, which pointed out the problem of merging clusters when optimizing modularity, Lancichinetti & Fortunato have recently found another drawback of modularity. It seems that the well known resolution limit is not only affecting the merging of small well defined communities, but it also presents a problem which is the splitting of large communities [29]. They claim that multiresolution modularity suffers from two opposite coexisting problems: the tendency to merge small subgraphs, which happens when the resolution is low; and the tendency to split large clusters, which dominates when the resolution is high. The authors claim that in benchmarks with heterogeneous distributions of cluster sizes, the simultaneous elimination of both biases is not possible and multiresolution modularity is not capable to recover the community structure. Fig 4.1 is a synthetic representation of a network formed by a random subgraph and two cliques, each joint to each other by a single link, where two different cluster sizes coexist. Indeed, multiresolution modularity optimization algorithms are suitable for discerning the internal structure of a network with

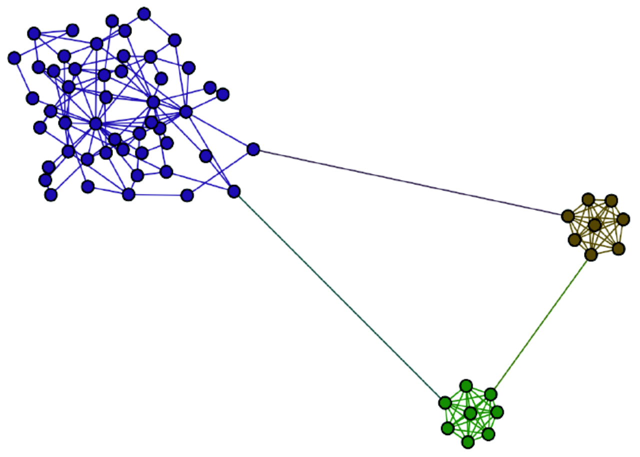

Figure 4.1: Schematic network with two cliques and a random subgraph, which are the natural communities of the network.

different scales of resolution, but it fails to find them at the same time. Say that we are using the AFG algorithm to detect communities on a network of this type. If the resistance parameter is set to a value such as to correctly identify the two cliques in different communities, then the random subgraph will also be divided into different communities, which does not coincide with the natural division of the network. Again, if the resistance is such that we identify the random subgraph as a single community, then the two cliques are merged into a single community.

There is a big difference between the original explanation of the resolution limit stated in section 2.4 and the one stated here, and therefore the way to solve them is also different. In the first case all nodes of the network shared the same structure. There was more than one solution to the clustering problem, as different levels of resolution coexisted in the same network, but all nodes participated in the same community structure. For this reason, the resolution limit could be diminished by screening the network at different resolution levels in order to unveil the whole structure. But in this new aspect of the resolution limit, the problem is slightly different: not all nodes have the same community structure. Instead, they form very different communities, in terms of configuration and size. Therefore, a single community detection analysis is not enough to discover the community structure, even if this analysis consists in screening different levels of resolution. The explanation is straightforward. Imagine we have a network consisting of two groups of densely connected nodes, very different in terms of size. As the communities sizes are very different, it is likely that they belong to different levels of resolution. As we are observing the whole network when performing the community analysis, when the resolution is the right one to observe the small community, this resolution is not adequate for the observation of the big community, and viceversa. The solution would then be to analyze the network in different parts, each one with its own resolution parameter, in a hierarchical way, which is the solution presented in the following section.

4.2 Hierarchical AFG solution to the resolution limit

As stated before, this section presents the refinement of the AFG algorithm that makes it suitable to use even in situations where the problem of the resolution limit arises. This algorithm provides insight of the community structure present in a network when different levels of resolution coexist. The new approach to solve the resolution problem takes advantage of the capability of the AFG method to find meaningful communities from the initial steps of the mesoscale analysis. More precisely, we propose the use of an iterative scheme which combines the optimization of modularity close to the macroscale of the network with its splitting in subgraphs, one for each of the previously found communities.

Supposing that our network is undirected, weighted, with positive weights and no self-loops, the prescription of our algorithm is the following:

- Start out from the macroscale partition $\mathcal{M}$, which has only one community containing all nodes. Then, find the upper bound of this macroscale, which is the minimum value of the resistance parameter (r_{min}) needed to find a partition $\mathcal{C}$ of the network with optimal modularity $Q_{\text{AFG}}[w_{ij}, \mathcal{C}, r_{\text{min}}]$ formed by more than one community.
- Split the network in the subgraphs defined by the partition $\mathcal{C}$ just found.
- Repeat the previous steps with each subgraph until no further subdivisions are needed.

To illustrate this process, let us take the synthetic benchmark in Fig. 4.1 as an example. Starting out from the partition containing all nodes, we have to find the value of r that gives us an optimal partition in terms of modularity. This partition is expected to divide the network in two communities: the random subgraph and the two cliques together. Then, we separate the two communities in different subgraphs (that is, cutting the links between those communities) and perform the same analysis for each subgraph. When dividing the random subgraph, one would expect to find a non-informative partition, and the same happens with the further divisions of the new communities. However, when dividing the community containing the two cliques, what should happen is that the partition found consists in two communities of a clique each. Then the question is the following:

how to distinguish between those divisions in the network that are relevant from the ones that are not? The criterion will be the same as in the original AFG method, which is that partitions that remain unchanged for a longer period of the resistance parameter should be the most relevant ones. With the hierarchical AFG the algorithm defines a hierarchical organization of the nodes, where the values of $r_{\min}$ at each splitting define the ultrametric distances between nodes, i.e. the heights in the dendrogram at which every pair of nodes first meet.

4.2.1 Fast-Tracking Resistance algorithm

The hierarchical algorithm presented includes a method designed to find the exact value of r at which each subgraph splits for the first time. In fact, the calculations for $r_{\min}$ and the partition with optimal modularity with more than one community ($\mathcal{C}$) are performed at the same time, therefore avoiding the costly scanning of the whole mesoscale between the lower and upper bounds of the resistance. This a consequence of the following properties.

- The value of $r_{\min}$ is negative, with the only exception in which the network is just a clique.
- $Q_{\text{AFG}}[w_{ij}, \mathcal{M}, r] = 0$, $\forall r < 0$, because:

$$\begin{aligned} Q_{\text{AFG}}[w_{ij}, \mathcal{M}, r] &= \frac{1}{2w + N|r|} \sum_i \sum_j \left[w_{ij} + r\delta_{ij} - \left(\frac{w_i w_j}{2w} - \frac{r^2}{N|r|} \right) \right] \\ &= \frac{1}{2w + N|r|} \left[2w + Nr - \left(\frac{(2w)^2}{2w} + \frac{N^2 r^2}{Nr} \right) \right] = 0 \quad (4.1) \end{aligned}$$

  In fact, modularity Eq. 2.19 is always zero for $\mathcal{M}$, no matter the network or the value of the self-loops.
- Since $Q_{\text{AFG}}[w_{ij}, \mathcal{M}, r] = 0$ and modularity is a continuous and monotonically increasing function of the resistance for any given $C \neq \mathcal{M}$, the optimal partition $\mathcal{C}$ at $r_{\min}$ must satisfy $Q_{\text{AFG}}[w_{ij}, \mathcal{C}, r_{\min}] = 0$.
- For any given partition C, the minimum meaningful value of the resistance $r_{\min}(C)$ is the one for which $Q_{\text{AFG}}[w_{ij}, C, r_{\min}(C)] = 0$. Thus, Eq. (4.1) leads to

$$r_{\min}(C) = \frac{-2w}{N - \frac{1}{N} \sum_{s \in C} n_s^2} Q[w_{ij}, C] \, . \tag{4.2}$$

- The upper bound of the macroscale is given by

$$r_{\min} = \min_C \{ r_{\min}(C) \} \tag{4.3}$$

and $\mathcal{C}$ is the partition which minimizes $r_{\min}(C)$.

All these properties may be combined in the following *fast-tracking resistance* (FTR) algorithm to find the upper bound of the macroscale:

1. Optimize modularity at $r = 0$, to obtain partition C_{prev}.
2. Calculate $r_{\min}(C_{\text{prev}})$ using Eq. (4.2).
3. Optimize modularity at $r := r_{\min}(C_{\text{prev}})$, to obtain the current partition C_{curr}.
4. If $C_{\text{curr}} = C_{\text{prev}}$ or $C_{\text{curr}} = \mathcal{M}$, then $r_{\min} := r_{\min}(C_{\text{prev}})$ and $\mathcal{C} := C_{\text{prev}}$.
5. Otherwise, let $C_{\text{prev}} := C_{\text{curr}}$ and go back to the second step.

In practice, this algorithm converges in a few number of steps. It stops when a value of r is found such that the optimization of modularity does not produce any new partition. In this case, the modularity of both C_{prev} and $\mathcal{M}$ is zero, and no known partition can be used to obtain a better upper bound of the macroscale. Of course, we cannot claim that we have found the "real" $r_{\min}$, since no optimization heuristic can ensure the finding of the global maximum of modularity (this problem is known to be a NP-hard problem, see [10]), but this is the best approximation one may obtain.

4.2.2 Application and performance of the Hierarchical AFG

In this section we will apply the Hierarchical AFG algorithm to a benchmark presented by Fortunato & Lancichinetti [29] specially designed to show that multiresolution algorithms fail to overcome the resolution limit. Our intention is to show that the hierarchical version of the AFG algorithm is capable to find the natural configuration of communities and that this solution is also the most persistent throughout the screening of the mesoscale. The benchmark used consists in a network formed by three natural communities: the first one is an Erds-Renyi (ER) network formed by 400 nodes, and the others are two cliques of 13 nodes each, all three connected between them by a single link. A representation of the network is shown in Fig. 4.2. Although the difference of resolutions can be seen from this representation at bare eye, there exists a mechanism to determine if a network will present resolution problems or not. The way to do so is to plot a link density map and detect if there are sharp contrasts. It is expected that if very different topological scales coexist, there will also be jumps in the clustering coefficient. In Fig. 4.3 the clustering coefficient of each node is represented. Is it easy to see that nodes labeled from 1 to 400, which belong to the ER subgraph, have a low clustering coefficient, while those labeled from 400 to 426, which belong to the cliques, have a drastically higher clustering.

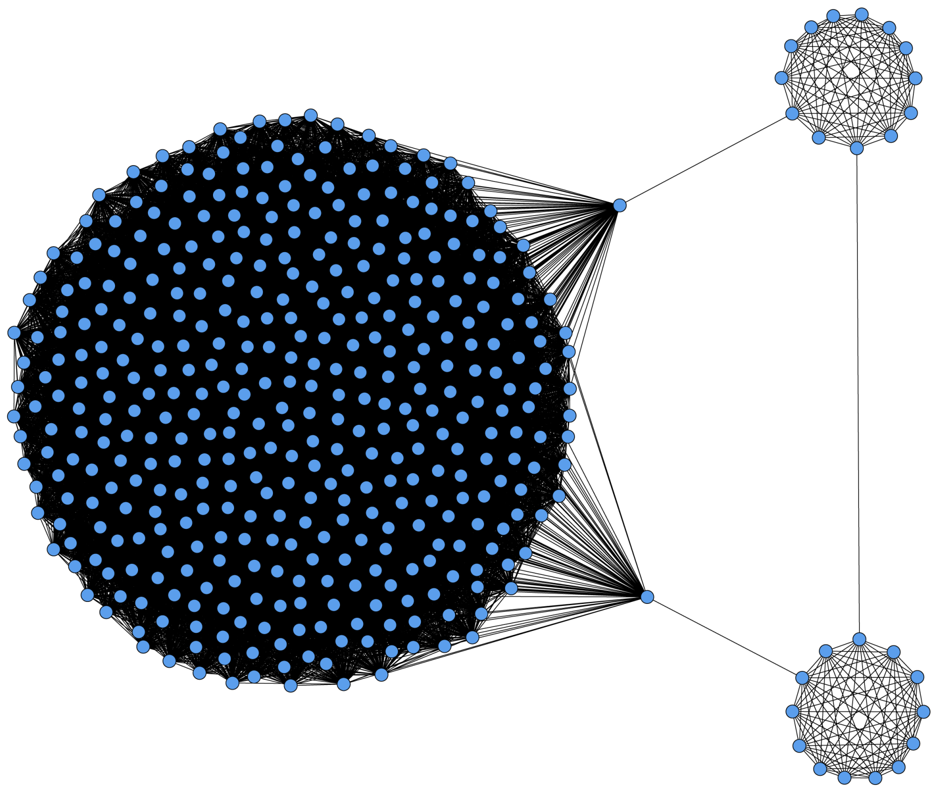

Figure 4.2: Benchmark proposed by [29] to test the resolution limit of multiresolution methods. The large component is a ER network of 400 nodes with $k = 100$ linked to two cliques of 13 nodes each, sharing only one link between them. The goal is to separate the three subgraphs using a community detection algorithm aimed to detect multiple resolutions.

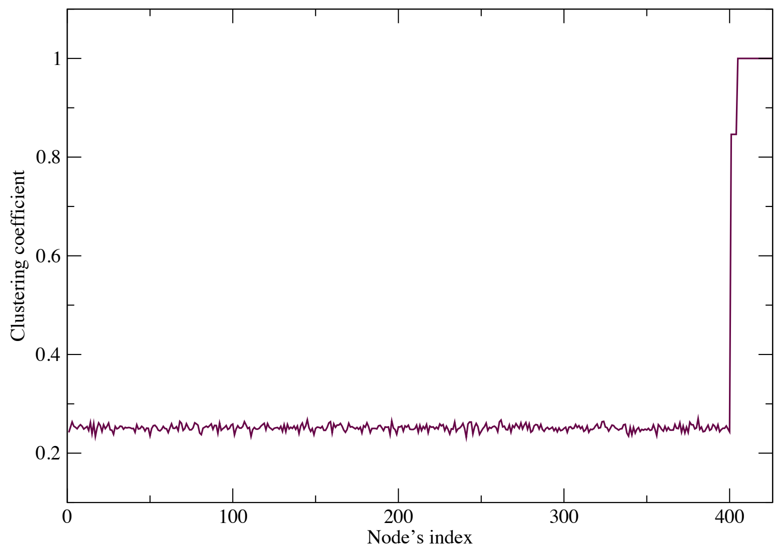

Figure 4.3: Clustering coefficient for the benchmark of Fig.4.2. Note the sharp transition in the relative local density of links represented by the clustering coefficient. This is indeed a hint to the coexistence of very different topological scales in the network.

The benchmark presented is indeed difficult to partition into the intuitive communities, even with multiresolution algorithms, as the author pointed out. The original AFG method failed to unveil the natural structure of the network due to the problem of analyzing all the network as a whole. However, next we show that with the hierarchical version this can be done with great success. If we calculate the upper bound of the macroscale with the Fast-Tracking Resistance algorithm, the value is $r_{\min} = -12.50$. For this value of the resistance, the partition obtained is formed by two communities formed by 400 and 26 nodes respectively, which correspond to the ER subgraph and the two cliques. Now, the algorithm separates those two communities cutting the links between them, and now each one forms a network by itself. Starting out from the network of 26 nodes, the value of $r_{\min}$ is -11.69, and the configuration obtained is a partition with two communities of 13 nodes each. If we now perform the same procedure for each clique, we will see that the partition containing all nodes together in the same community remains the best option in terms of modularity for the whole screening of r, and therefore these communities remain unchanged. As for the network formed by 400 nodes, the next value of $r_{\min}$ is -8.97, which gives us the intuitively non-informative partition that divides the network into two communities of 182 and 218 nodes. Following the analysis, these networks are further divided into new communities, until all nodes are isolated. The results obtained during this process can be more clearly visualized when represented in a dendrogram form, as shown in Fig. 4.4, in which the values of r at which every community is split are represented as the heights of the vertical lines. From this representation, is it easy to see that there exists a configuration of nodes that remain unchanged for more values of r than the rest. This happens when the network formed by 400 nodes is still not divided but the network formed by 26 nodes has already split into two groups, which is represented in the figure with a grey rectangle. A period of r longer than this one is not found during the rest of the dendrogram. With the automatic detection of this area, which is also included in the algorithm, the outcome of the algorithm would be the proper partitioning of the original network in the three natural communities that we were expecting to find, an encouraging result that reveals the Hierarchical AFG algorithm as capable to overcome the resolution limit of modularity in complex networks.

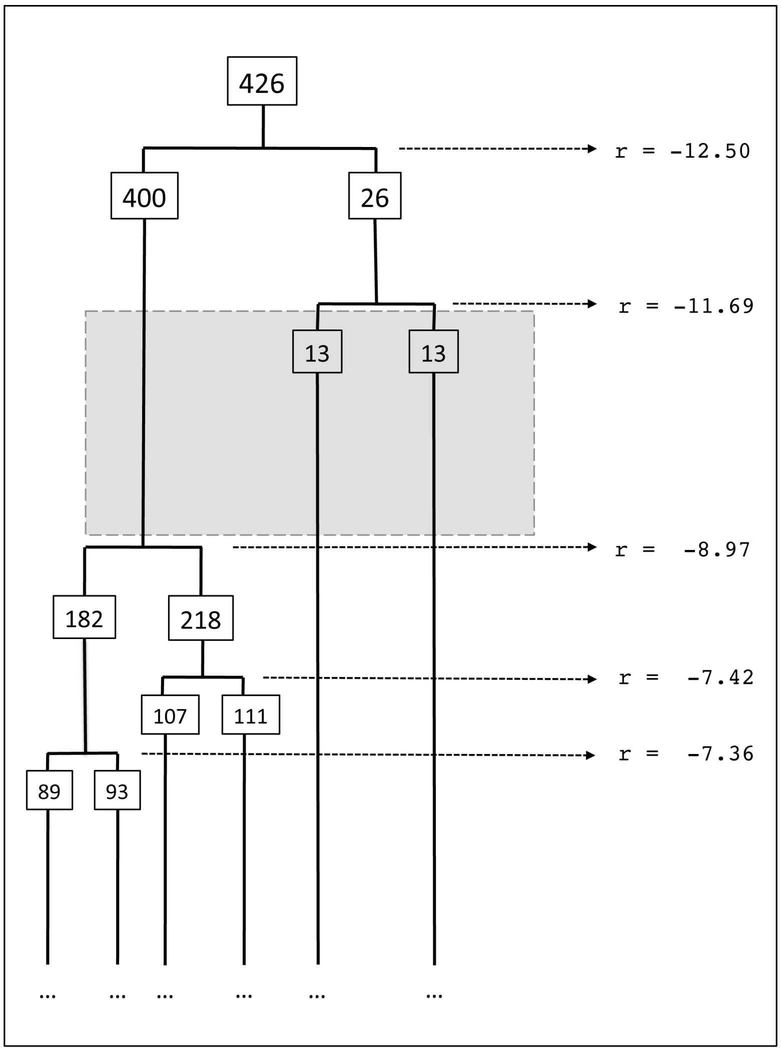

Figure 4.4: Dendrogram resulting from the application of the hierarchical multiresolution method on the benchmark of Fig. 4.2. The grey region shows the range of the resistance parameter in which the three communities searched coexist. Note that the vertical lines are not scaled.

Chapter 5

Conclusions and Future Work

In this document we have presented the work done for the master thesis, along with the theoretical foundations of the field of research in which the work is enclosed, complex networks. The original work developed along the thesis can be divided in three main contributions: i) Theoretical analysis to evolve the AFG method for weighted signed and directed networks, that can be found in Section 2.5.3; ii) Conceptual and practical mapping of data clustering problem into a community detection problem in networks; that corresponds to the whole Chapter 3; and finally, iii) Design of a new algorithmic scheme to overcome the resolution limit when using the AFG method, that corresponds to the whole Chapter 4.

The first contribution is relevant in complex networks because it will allow the mesoscopic analysis of networks in the most general case of weighted, signed, and directed adjacency matrices. With this approach we have been able to boost the computational capabilities of the AFG method, which is becoming a standard in the field and is on the scope of the new applications to time-varying networks.

The second contribution presented is the adaption of network analysis to a very important artificial intelligence problem: the data clustering problem. It has been shown that the mapping of a dataset to a complex network is possible and interesting as a new technique. For the case of the Iris dataset we have shown that the results obtained are competitive with other data clustering algorithms confronting the same problem.

The last work presented is not as focused to the applications of community detection methods, instead its purpose is to improve the AFG method in order to make it more resilient to the effects of the resolution limit, a major concern in the optimization of modularity, and we have shown its success in a synthetic network specially designed to prove the existence of the resolution limit.

Although the work presented is wide, there are still a lot of lines of research to be explored. The major goal from now on is to adapt the AFG and the Hierarchical AFG methods to the new challenges in complex networks, mainly time varying networks -networks where the nodes change their connections in time- and multilayer networks -networks where the links have different meanings depending on the categorical layer at which they belong-, as well as to design new algorithms to make community detection more efficient and reliable.

Appendix A

List of publications

Listed below are the publications related to the work presented in this document.

1. *Hierarchical multiresolution method to overcome the resolution limit in complex networks*, C. Granell, S. Gómez and A. Arenas, International Journal of Bifurcation and Chaos, 22, 1250171 (2012)
2. *Unsupervised clustering analysis: A multiscale complex networks approach*, C. Granell, S. Gómez and A. Arenas, International Journal of Bifurcation and Chaos, 22, 1230023 (2012)
3. *Mesoscopic analysis of networks: applications to exploratory analysis and data clustering*, C. Granell, S. Gómez and A. Arenas, Chaos: An Interdisciplinary Journal of Nonlinear Science, 21, 016102 (2011)
4. *Data clustering using community detection*, C. Granell, S. Gómez and A. Arenas, International Journal of complex systems in Science, 1, 21-24 (2011)

Bibliography

[1] A. Fernández A. Arenas, J. Duch and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9:176, 2007.

[2] R. Albert and A-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74:47, 2002.

[3] A. Arenas, A. Fernández, and S. Gómez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(053039), 2008.

[4] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] E. T. Bell. Exponential numbers. *Amer. Math. Monthly*, 41:411, 1934.

[6] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008.

[7] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D-U. Hwang. Complex networks : Structure and dynamics. *Phys. Rep.*, 424(4-5):175–308, 2006.

[8] S. Boettcher and A.G. Percus. Optimization with extremal dynamics. *Phys. Rev. Lett.*, 86:5211–5214, 2001.

[9] B. Bollobas. *Modern Graph Theory.* Springer, 1998.

[10] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Trans. Knowl. Data Eng.*, 20:172, 2008.

[11] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 1994.

[12] J. Duch and A. Arenas. Community identification using extremal optimization. *Phys. Rev. E*, 72(027104), 2005.

[13] P. Erdös and A. Rényi. On random graphs. I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[14] P. Erdös and A. Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[15] A. Fernández and S. Gómez. Solving non-uniqueness in agglomerative hierarchical clustering using multidendrograms. *J. Classification*, 25(1):43–65, 2008.

[16] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA*, 104:36, 2007.

[17] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. Series on Statistics and Applied Probability. ASA-SIAM, 2007.

[18] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821, 2002.

[19] F. Glover. Future paths for integer programming and links to artificial intelligence. *Comp. and Op. Res*, 5:533, 1986.

[20] S. Gómez, P. Jensen, and A. Arenas. Analysis of community structure in networks of correlated data. *Phys. Rev. E*, 80:016114, 2009.

[21] R. Guimerà and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895, 2005.

[22] B. A. Huberman. *The laws of the Web : patterns in the ecology of information*. MIT Press, Cambridge, Mass. :, 2001.

[23] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series. Prentice-Hall, Inc., 1988.

[24] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comp. Surv.*, 31(2), 1999.

[25] H. Jeong, B. Tombor, R. Albert, Z. N. Oltval, and A-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, October 2000.

[26] P. F. Jonsson, T. Cavanna, D. Zicha, and P. A. Bates. Cluster analysis of networks generated through homology: Automatic identification of important protein communities involved in cancer metastasis. *BMC Bioinf.*, 7, 2006.

[27] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 2005.

[28] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49:291–307, 1970.

[29] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, 2011.

[30] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[31] M. E. J. Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics*, 45(2):167–256, 2003.

[32] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, 2004.

[33] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.

[34] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577, 2006.

[35] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.

[36] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[37] R. Pastor-Satorras and A. Vespignani. *Evolution and Structure of the Internet: A Statistical Physics Approach.* Cambridge University Press, New York, NY, USA, 2004.

[38] J. M. Pujol, J. Béjar, and J. Delgado. Clustering algorithm for determining community structure in large networks. *Phys. Rev. E*, 74(016107), 2006.

[39] J. Reichardt and S. Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Phys. Rev. Lett.*, 93(218701), 2004.

[40] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, 2006.

[41] P. Schuetz and A. Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Phys. Rev. E*, 77(4), 2008.

[42] M. A. Serrano and M. Boguñá. Topology of the world trade web. *Physical Review E*, 68:015101(R), 2003.

[43] V.A. Traag and J. Bruggeman. Community detection in networks with positive and negative links. *Phys. Rev. E*, 80(036115), 2009.

[44] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Number 8 in Structural analysis in the social sciences. Cambridge University Press, 1 edition, 1994.

[45] D. J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company, 1st edition, 2003.

[46] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[47] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks In Neural Networks*, 16(3):645–678, 2005.

[48] W.W. Zachary. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.*, 33:452–473, 1977.