UNIVERSITÀ DELLA CALABRIA

Dipartimento di Elettronica,
Informatica e Sistemistica

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXVciclo

Settore Scientifico Disciplinare: ING-INF/05

Tesi di Dottorato

Pattern Extraction from Data with application to Image Processing

Alessia Amelio

Università della Calabria

Dipartimento di Elettronica,
Informatica e Sistemistica

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXV ciclo

Tesi di Dottorato

Pattern Extraction from Data with application to Image Processing

Alessia Amelio
[illegible]

Coordinatore
Luigi Palopoli [illegible]

Supervisori
Luigi Palopoli [illegible]
Clara Pizzuti [illegible]

DEIS

DEIS- DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA
Novembre

Settore Scientifico Disciplinare: ING-INF/05

To Anyone interested in reading this work

Preface

This thesis describes my research activities during the Ph.D. course, including all my research work developed in these three years.

It has been a long way since I decided to attend the Ph.D. Along this road, many things have changed my point of view and improved my personal knowledge, not only about Computer Science. In particular, I've appreciated very much my experiences at Georgia Institute of Technology in Atlanta, USA and at ICAR, National Research Council of Italy (CNR) in Rende (CS), Italy. This is mainly because I have spent precious time there to learn about my favourite research topics.

And now that the end of my Ph.D. is close, I'd like to express my total gratitude to my supervisors who always encouraged my ideas and who supported my work with dedication.

Finally, it is my greatest desire that the developed approaches in image processing and presented in this thesis may be used only for peaceful purposes.

Alessia Amelio
Signature
[illegible]

Contents

Abstract

The term *Information Extraction* refers to the automatic extraction of structured information from data. In such a context, the task of pattern extraction plays a key role, as it allows to identify particular trends and recurring structures of interest to a given user. For this reason, pattern extraction techniques are available in a wide range of applications, such as enterprise applications, personal information management, web oriented and scientific applications. In this thesis, analysis is focused on pattern extraction techniques from images and from political data. Patterns in image processing are defined as features derived from the subdivision of the image in regions or objects and several techniques have been introduced in the literature for extracting these kinds of features. Specifically, image segmentation approaches divide an image in ”uniform” region patterns and both boundary detection and region-clustering based algorithms have been adopted to solve this problem. A drawback of these methods is that the number of clusters must be predetermined. Furthermore, evolutionary techniques have been successfully applied to the problem of image segmentation. However, one of the main problems of such approaches is the determination of the number of regions, that cannot be changed during execution. Consequently, we formalize a new genetic graph-based image segmentation algorithm that, thanks to the new fitness function, a new concept of neighborhood of pixels and the genetic representation, is able to partition images without the need to set a priori the number of segments. On the other hand, some image compression algorithms, recently proposed in literature, extract image patterns for performing compression, such as extensions to 2D of the classical Lempel-Ziv parses, where repeated occurrences of a pattern are substituted by a pointer to that pattern. However, they require a preliminary linearization of the image and a consequent extraction of linear patterns. This could miss some 2D recurrent structures which are present inside the image. We propose here a new technique of image compression which extracts 2D motif patterns from the image in which also some pixels are omitted in order to increase the gain in compression and which uses these patterns to perform compression. About pattern extraction in political science, it consists in detecting voter profiles, ideological positions and political interactions from political data. Some proposed pattern extraction techniques analyze the Finnish Parliament and the United States Senate in order to discover political trends. Specifically, hierarchical clustering has been employed to discover meaningful groups of senators inside the United States Senate. Furthermore, different methods of community detection, based on the concept of modularity, have been used to detect the hierarchical and modular design of the networks of U.S. parliamentarians. In addition, SVD has been applied to analyze the votes of the U.S. House of Representatives. In this thesis, we analyze the Italian Parliament by using different tools coming from Data Mining and Network Analysis with the aim of characterizing the changes occurred inside the Parliament, without any prior knowledge about the ideology or political affiliation of its representatives, but considering only the votes cast by each parliamentarian.

1 General introduction

Information Extraction is defined as the automatic extraction of structured information (entities, relationships between entities and attributes) from data. The type of data used as origin can be structured, such as for relational data from databases, semi-structured, such as HTML or XML, and unstructured (i.e. plain text or images). The extraction process also allows to integrate structured and unstructured data sources and to perform queries on them.

Pattern Extraction is the main task in information extraction. It consists in identifying the patterns which are relevant based on the user's queries. Such patterns can be detected automatically, semi-automatically and manually. In particular, the pattern discovery automatic procedures support the user in building the knowledge-bases, taking a set of input data and extracting sets of patterns from them by using various kinds of learning techniques.

The topic of pattern extraction is relevant in multiple communities including machine learning, information retrieval, database, web, image processing and document analysis. In the past, the extraction process was focused on the identification of named entities, like people and company names and relationship among them from natural language text. In the last two decades, the introduction of the Internet considerably promoted the diffusion of applications based on various forms of information extraction. Consequently, new applications of pattern extraction have been adopted, because of the easy online access to both structured and unstructured data. Furthermore, applications such as comparison shopping, and other automatic portal creation applications, have increased the research and the commercial activity on this topic.

The techniques of pattern extraction developed over the last years were initially rule-based with manually coded rules. However, because manual coding of rules quickly becomes hard, algorithms for automatically learning rules from examples were subsequently introduced. As the extraction systems began to be used on more noisy unstructured sources, rules were not considered to be suitable. Consequently, statistical learning techniques were adopted, in particular generative models based on Hidden Markov Models and conditional models based on maximum entropy. Afterwards, both these approaches were substituted by global conditional models, called Conditional Random Fields. Furthermore, when the analysis of a documents structure was required from the extraction systems, techniques from grammar construction were introduced. Finally, hybrid models have been adopted, trying to embody the benefits of both statistical and rule-based methods.

Pattern extraction is currently used in a wide range of applications, such as enterprise applications, personal information management, web oriented and scientific applications.

About enterprise applications, a typical example of pattern extraction is automatically keep track of specific kinds of events from news sources. Some tasks consist of the extraction of structured entities like people and company names, and relations between them. Other popular tasks are based on tracking disease outbreaks and terrorist events from news sources. Furthermore, the automatic creation of multimedia news by the integration of video and pictures of entities and events annotated in the news articles and the hyperlink of news articles to background information on people, places and companies are two recent applications of pattern extraction on news articles. Other examples of enterprise applications can be found in customer-oriented enterprises. A customer-oriented enterprise has to manage many types of unstructured data coming from the customer interaction, which must comply with the structure of the enterprise's databases and business ontologies. Some tasks in this direction are based on the identification of special patterns, such as product names and product attributes from customer emails and links between customer emails and a given transaction in a sales database, on the extraction of patterns like merchant name and addresses from sales invoices, on the identification of customer moods from phone conversation transcripts, and on the discovery of product attribute value pairs from textual product descriptions. Furthermore, in data warehousing, cleaning processes require to convert addresses that are stored as flat strings, whose structure is little explicit, into their structured forms such as road name, city, and state. Sometimes, for the same person, different address records are stored in multiple databases. When the warehouse is built, all these addresses must be converted in a standard canonical format where all the different patterns representing the fields are identified and duplicates eliminated. Finally, an implicit data structure is also present in classified ads and other listings such as restaurant lists. These kinds of data can be invaluable for querying if a structured format is not given by extracting from them the most meaningful patterns.

About Personal information management (PIM) systems, the goal is to organize personal data such as documents, emails, projects and people in a structured inter-linked format. Basically, such systems should be able to automatically detect structured patterns from existing file-based unstructured sources.

Regarding pattern extraction in web oriented applications, different examples can be given about. The first one is related to citation databases on the web that need a complex structure extraction process from sources such as conference web sites or individual home pages (Citeseer, Google Scholar3, Cora, etc.). Pattern extraction for such databases is composed of multiple levels starting from navigating web sites for finding pages containing publication records, extracting individual publication records from a HTML page, extracting title, authors, and references from paper PDFs, and segmenting citation strings into individual authors, title, venue, and year fields. A second example is related to opinion databases. Many web sites store unmoderated opinions about a variety of topics, such as products, books, movies, people, and music, which are in free text form inside blogs, newsgroup posts, review sites, etc. If these opinions are organized along structured fields identified by a pattern extraction process, their value can increase. Community web sites such as DBLife and Rexa are other examples of structured database creation from web documents. They keep information about researchers, conferences, talks, projects, and events related to a specific community. The main steps for creating such structured databases consist of locating patterns such as talk announcements from department pages, extracting patterns like names of speakers and titles from them, structured records about a conference from a website, and so on. Furthermore, recently much effort has been done in creating comparison shopping web sites that automatically crawl merchant web sites to find products and their prices, for comparison shopping. Advertisement placement on webpages and structured web searches are other two important contexts where pattern extraction processes take place. For example, mentions of products and the type of opinion expressed on the products can be extracted from web sites to place advertisements.

Finally, pattern extraction process is of prior importance in many scientific fields, such as bio-informatics, image processing, computational political science, and so on. In bio-informatics, the detection of ”biological objects” such as proteins and genes is a meaningful task of pattern extraction. Another central problem consists of extracting from paper repositories like Pubmed, protein names, and their interaction. This task favored the development of new computational techniques used for pattern extraction. It is mainly because the entities in biological field like Gene and Protein names are very different from classical named entities, such as people and companies in enterprises context.

This thesis focuses of techniques for features extraction, specifically covering the two application domain of pattern extraction from images and political data. Such two application domains are briefly overviewed upon next.

1.1 Pattern extraction from images

Patterns in image processing can be considered as features derived from the subdivision of the image in ”uniform” regions or objects, features extracted from the entire image or just regular sub-area of an image or features which are dependent from the application, such as human faces and fingerprints.

Object pattern extraction in images can be performed by using various techniques, such as edge detection and image segmentation. An edge is considered as a boundary between two dissimilar regions in an image, which may indicate different parts of an object, or also a boundary tracing the separation between light and shadow on a single surface. Most edge detection algorithms assume that a discontinuity in the image intensity function or a very sharp intensity gradient in the image are able to identify an edge. About image segmentation, it is a process whose aim is to partition a digital image into disjoint and uniform regions. Specifically, this process should be able to group pixels together into regions based on some similarity criterion.

Sometimes it is important to extract from images specific kinds of patterns such as textures, corners and point patterns.

Textures are considered as image patterns with properties of homogeneity that do not consist in the presence of a single color or intensity. Texture is a property which characterizes all surfaces, including cloud, trees, bricks, hair, fabric, etc. and codifies important information about their structural arrangement and their relationship to the surrounding environment. Most texture patterns can be classified as random or periodic. In this context, pattern extraction procedures are based on texture segmentation, whose aim is the identification of disjoint regions inside the image based on their texture uniformity. In this context, segmentation is often performed by adopting three independent subprocesses: texture feature extraction, feature selection or reduction if the number of features is too large and, finally, application of a segmentation algorithm.

A corner is another important kind of two-dimensional pattern localized inside an image, which can be characterized by using a mathematical description. It is the feature formed at boundaries between only two image brightness regions, in correspondence to the ”extremely” high values of the boundary curvature. Image corner detection is a well known task in computer vision and image processing. Standard approaches for corner detection consist in a pre-segmentation phase for contours detection and a subsequent extraction process of corners corresponding to the intersection points or junction points between straight lines.

Finally, point patterns are defined as point sets with finite diameters which are present in various pattern recognition and image processing tasks. The points may be defined as feature vectors in feature space, pixels in images, physical objects or even spatial data. The structure in the point set is usually discovered in the form of clusters, directionality, and intrinsic dimensionality. Generally, it is important to discover the border, known as the external shape, of a point set.

Another interesting task in image pattern extraction is related to the geometric primitive discovery. In particular, the surface of the objects inside the image can be defined as a set of patches, where each patch is a geometric primitive. Consequently, geometric primitive location and discovery is important for identifying and locating object patterns in the environment. It should evaluate if the set of points can be best fitted by the given geometric primitive or not. On the other hand, the extraction process of geometric primitives should find the right primitive that fits the set of points.

1.2 Pattern extraction from political data

In computational political science, voter profiles, ideological positions, political interactions and the content of political conflict are considered as meaningful patterns that should be discovered from data. In this context, pattern extraction has been performed by using several computational techniques, including text analysis methods, social network analysis and agent-based models [82].

About text analysis, data mining techniques are often adopted, in order to discover high-quality information like patterns and trends in politics from text automatically. Several proposed techniques generate word scores from reference texts with a priori known positions, and then score each virgin text by the generated word scores. Given the features of words, usually the next phase consists in categorize the texts by using the word features through classification and clustering approaches. Sentiment analysis is another branch of text analysis used in political science, whose aim is to detect patterns like the attitudes of speakers or writers with respect to some topic in text. Standard sentiment analysis should analyze the words from the linguistic point of view, i.e. studying the sentiments on words and phrases and discovering the emotional bias of key terms. Probabilistic Topic Models are also employed for pattern extraction from political text. They consider documents as mixtures of topic patterns, which are probability distributions over words. A topic model is a generative model for documents, which is a simple probabilistic procedure to generate the documents. An example is given by the topic modeling of the United States Senate speeches. Topic patterns extracted from the data are then labeled to build meaningful classes.

The analysis of interactions, lives, competitions and cooperations among political subjects is another means to discover meaningful patterns in political data. In particular, social networks have been adopted for this purpose. They are defined as social structures composed of nodes representing all the main actors in a political world, and of edges connecting pairs of nodes, representing specific kinds of relations among them. Traditional network measures from graph theory and exploratory analysis, such as closeness, betweenness, and eigenvector centrality, have been employed to analyze these kinds of networks, compute statistics over the implied graphs, and discover the most central political interest groups. Social network analysis techniques can also be used to analyze multiple forms of political co-operation.

Finally, in order to simulate political evolution patterns in terms of policy consequences, election outcomes and trade changes, agent-based approaches represent also useful methods. In particular, parties are modeled as rationally bounded adaptive agents, and multiple algorithms are adopted to represent different behaviors of parties to explore election results.

1.3 Thesis plan

In this section we have just presented the variety of aspects and communities involving the information extraction topic, with particular emphasis on the task of pattern extraction. In such context, we have provided specifically an overview of the main aspects related to image pattern extraction approaches and pattern extraction procedures in computational political science, since the rest of this thesis will be focused on pattern extraction applications in image processing and political science. The thesis is organized as follows. In Part I we will introduce two new techniques respectively for image compression and image segmentation that employ pattern extraction procedures. Segmentation (Section 2) is obtained by an evolutionary algorithm and it is used as a mean to extract ”uniform” region patterns, while compression (Section 3) is performed by discovering motif patterns in two dimensions. Section 4 will conclude Part I.

Part II of this thesis will analyze some aspects related to kinds of patterns extracted from political data. In particular, Section 5 will introduce a new approach that employs social network analysis and data mining for characterizing trends, alliances, modifications and conflicts incurred inside the Italian Parliament during the XVI legislature and Section 6 will conclude Part II. Finally, Section 7 will summarize all the presented approaches and will outline future work.

Part I

Summary

Part I of this thesis will present two new image processing approaches where pattern extraction plays a relevant role. The first one is an image segmentation algorithm which extracts meaningful region patterns from natural images while the second one is an image compression technique based on the extraction of repeated ”not solid” motif patterns.

Specifically, the first technique consists in a graph-based approach for image segmentation that employs genetic algorithms. An image is modeled as a weighted undirected graph, where nodes correspond to pixels, and edges connect similar pixels. A fitness function, that extends the normalized cut criterion, is employed, and a new concept of nearest neighbor, that takes into account not only the spatial location of a pixel, but also the affinity with the other pixels contained in the neighborhood, is defined. Because of the locus-based representation of individuals, the method is able to partition images in ”uniform” region patterns that well adhere to the human visual perception, without the need to set the number of segments beforehand.

The image compression algorithm is based on extensions to 2D of some of the Lempel-Ziv incremental parsing techniques. In these techniques, an image is decomposed into a number of patches, consisting each of a square or rectangular solid block. In this new approach, image compression techniques are proposed based on patches that are not necessarily solid blocks, but are affected instead by a controlled number of undetermined or *don’t care* pixels. Such patches are chosen from a set of candidate *motifs* that are extracted in turn from the image *2D motif basis*, the latter consisting of a compact set of patterns that result from the autocorrelation of the image to itself. As is expected, it is found that limited indeterminacy can be traded for higher compression at the expense of negligible loss.

2

An Evolutionary and Graph-based method for Image Segmentation

2.1 Introduction

Next we present a new graph-based algorithm, named *GeNCut* (*Genetic NCut*), to solve the image segmentation problem by using an evolutionary approach. In particular, we represent an image as a weighted undirected graph. Then, a genetic algorithm optimizing a fitness function is executed in order to find an optimal partitioning of the graph and, consequently, a good segmentation of the image. The fitness function is an extension of the normalized cut concept of Shi and Malik [65] that allows for a simultaneous k-way partitioning of the image without the need of fixing the number k of divisions beforehand, which is typical of many image segmentation approaches. In fact, because of the locus-based representation of individuals adopted, k is automatically determined by the optimal value of the objective function. Experiments on images of different difficulty show that *GeNCut* outperforms the method of Shi and Malik by partitioning natural and human scenes in meaningful objects.

The chapter is organized as follow. In the two next sections the problem of image segmentation is defined together with some notions of evolutionary computation. Then a detailed review of the state of the art on image segmentation is presented also contrasting our approach against to previous ones. After that, section 2.5 provides an informal overview of our technique. $GeNcut$ is formally described starting from section 2.6.1 that introduces a formalization of the image segmentation as a graph partitioning problem and a description of the adopted homogeneity measure. Section 2.6.2 presents the concept of normalized cut and the fitness function used by *GeNCut*. Section 2.6.3 explains the employed genetic representation and operators. Finally, section 2.7 illustrates the experimental results.

2.2 The image segmentation problem

Image segmentation is an important problem in pattern recognition that aims at partitioning an image into uniform and meaningful regions [18]. More formally, the problem can be stated as follows [28]: let R be an image constituted by a set of pixels. Segmenting the image R consists in subdividing R into a finite number of regions $R_1 ... R_s$ such that

1. $R = \bigcup_{i=1}^{s} R_i$.
2. R_i is a convex set, $i = 1 ... s^2$.
3. $R_i \cap R_j = \emptyset, \forall i, j, i \neq j$.
4. $Q(R_i) = TRUE, i = 1 ... s$.
5. $Q(R_i \bigcup R_j) = FALSE$, for each pair of adjacent regions R_i, R_j.

In this case, $Q(R_k)$ is a logical predicate defined on all the points of the set R_k, which is able to specify the existance of a given property on the set of the points. Two regions R_i and R_j are defined adjacent is their union is a connected set.

Condition 1 implies that the segmentation have to be complete: each pixel must belong to some region. Condition 2 requires that the points inside a region have to be connected (4 or 8-connected). Remember that, given two points P and W belonging to a same subset $S \subset R$, we can assert that P is k-connected to W if a k-path between them exists including only the points in S. A path of length n from P to W is a sequence of points $P = P_0, P_1, P_2, ...P_n = W$ such that P_i is a k-neighbor of $P_{i-1}, \forall i = 1...n$. A point P of position (x, y) in R has k=4 neighbors (of position $(x, y-1)$, $(x, y+1)$, $(x+1, y)$, $(x-1, y)$) and k=8 neighbors (of position $(x-1, y-1)$, $(x, y-1)$, $(x+1, y-1)$, $(x-1, y)$, $(x+1, y)$, $(x-1, y+1)$, $(x, y+1)$, $(x+1, y+1)$).

Condition 3 indicates that the regions have to be disjoint. Condition 4 is related to the property Q that should be satisfied by pixels inside a segmented region R_k. Finally, the condition 5 indicates that two adjacent regions R_i and R_j should be different considering the Q predicate. As an example, a homogeneity measure could be defined over pixels that takes into account characteristics such as intensity, color, or texture. Pixels belonging to the same region should be similar on the basis of the homogeneity measure adopted, while adjacent regions should be significantly dissimilar with respect to the same features.

Usually, image segmentation is an initial and vital step in a series of processes aimed at understanding an image. Applications of image segmentation include identifying objects in a scene for object-based measurements such as size and shape. It is a very important aspect in medical imaging for locating tumors and other pathologies, measuring tissue volumes, performing computer-guided surgery, diagnosis, treatment planning and study of anatomical structures. It refers also to satellite image segmentation where the main objects of an earth observation scene should be extracted and to *face recognition* where the initial step is the extraction of facial features from a human image. Also, identifying objects in a moving scene is a useful task in video-surveillance, while identifying objects which are at different distances from a sensor using depth measurements from a laser range finder is important in path planning for a mobile robot.

2.2.1 Quality evaluation for image segmentation

An image segmentation algorithm should be able to solve the segmentation problem, running until the regions or objects of interest have all been identified into the image. Consider that segmentation process is in general one of the most difficult goals to achieve in image processing and the segmentation accuracy determines the success or the fail of the image segmentation procedures. However, accuracy evaluation of segmentation algorithms remains today a critical point: this is mainly because there is no unique ground-truth segmentation of an image against which the output of an algorithm may be compared. Consequently, there is no standard method for objective evalutation of segmentation quality. Common approaches for assessing segmentation results are based on human intuition or judgment (*subjective evaluation*) [10]: an ”ad hoc” subjective evaluation is performed by a representative group of observers. To avoid systematic subjective evaluation, an automatic procedure, referred to as *objective evaluation*, is often adopted. Quality metrics used for evaluating objectively the image segmentation may consider segmentation algorithms or segmentation results. The metrics are referred to as analytical methods or empirical methods, respectively. Analitical methods don't implement the algorithms but evaluate segmentation algorithms by considering their principles, their requirements and their complexity. Empirical methods, on the other hand, assess the segmentation quality by analyzing the results of the segmentation algorithms. In order to compare segmentation algorithms based on empirical evaluation, the different approaches are applied on a set of test data that are relevant to a given application. The algorithm producing the best quality score is then chosen for that application.

Next, we discuss the evolutionary computation and then analyze the application of evolutionary computation to the image segmentation problem.

2.3 The evolutionary computation

Similarly to [21], evolution can be defined as a two-phase iterative process, consisting in a random variation and a selection phase. Similarly to the natural evolution beginning from an initial population, the evolutionary computation starts by selecting a set of initial contending solutions for a particular problem. These solutions can be randomly chosen or selected using any available knowledge about the problem . Then, the ”parent” solutions are joined by a preselected means of random variation, generating ”offspring”. A fitness function is able to evaluate the effectiveness of the ”children” solutions: those solutions whose fitness value is quite low are removed and consequently not further considered. So, the process is repeated over successive generations. One of the advantages and differences of the evolutionary algorithms from the other approaches is that they do not require many assumptions on how to evaluate the fitness of a solution (i.e. the gradient-based search requires a smooth, differentiable cost function). Basically, the performance index need only to compare two solutions to each other, determining which solution is better than another. For this reason, evolutionary algorithms can be used for the resolution of a broad range of non conventional problems. Other two advantages of the evolutionary computation are the possibility to ”on the fly” generate quickly good enough solutions ready to use and the property to adapt dynamically the optimization procedure, without restarting.

In an evolutionary procedure it’s very important to define, first of all, the format of the possible solutions that should be evaluated. Generally, there is no single best choice for this representation. Secondly, the fitness function, that is necessary to assess any candidate solution, must also be selected. The third step consists in choosing a random variation operator (or operators) useful to generate offspring solutions from parent solutions. The fourth step should establish a rule for choosing which solutions will survive for the next generation. One of the many forms of selection is survival of the very best solutions related to the fitness value. On the other hand, remaining solutions are thrown away. Finally, the initial population have to be selected. The solutions can be chosen completely at random from the space of all possible solutions if nothing is known about how to solve the problem. Alternatively, prior knowledge or bias of other algorithms for good solutions can be used to generate some reasonable start-points to incorporate in the initial population. Those solutions will survive in the evolution if they will prove useful; otherwise they will be removed as weak solutions.

In conclusion, the main point is that the problem-specific knowledge available is not necessary to the application of an evolutionary technique, although it is possible to integrate and take advantage of it in the evolutionary computation. This is the reason why evolutionary algorithms can be applied for the resolution of a broad range of problems.

2.3.1 Genetic algorithmic approach

Genetic Algorithms [26] are a class of adaptive general-purpose search techniques included in evolutionary computation and consequently inspired by natural evolution. They have been proposed by Holland [33] in the early 1970s as computer programs that simulate the evolution process in nature. In the last few years genetic algorithms revealed competitive alternative methods to traditional optimization and search techniques and they have been applied to many problems in diverse research and application areas such neural nets evolution, planning and scheduling, machine learning and pattern recognition. A standard Genetic Algorithm (GA) evolves a constant-size population of elements (called $chromosomes$) by using the genetic operator of *reproduction*, *crossover* and *mutation*. Each chromosome represents a candidate solution to a given problem and it is associated with a *fitness value* that reflects how good it is, with respect to the other solutions in the population. Generally, a chromosome is encoded as a string of bits from a binary alphabet. The reproduction operator copies elements of the current population into the next generation with a probability proportionate to their fitness (this strategy is also called roulette wheel selection scheme). The crossover operator generates two new chromosomes by crossing two elements of the population selected proportionate to their fitness. The mutation operator randomly alters the bits of the strings. GAs are stochastic iterative algorithms without converge guarantee. Termination may be triggered by reaching a maximum number of generations or by accepting a solution that satisfies some goodness criterion. GAs can be applied when no domain knowledge is available. However, domain knowledge can be incorporated in GAs by either modifying the genetic operators, or choosing a particular initial population, or modifying the quality function.

In the next section, we come back to describe the state of the art in image segmentation. It includes also an overview of the main evolutionary-based image segmentation techniques.

2.4 State of the art in image segmentation

The image segmentation problem has been intensively investigated with the use of several computational techniques, and many different methods have been proposed. A broad classification divides the existing methods in two main categories [77]: boundary detection-based approaches and region clustering-based approaches. The former approaches search for closed boundary contours by detecting pixels that sensibly change in intensity. Boundaries of objects are obtained by linking such pixels in contours. The main limitation of these approaches is that a threshold value must be set in order to produce a continuous contour [60, 31].

Region cluster-based methods group similar closed pixels into clusters. Many of these approaches use Fuzzy C-means [13] or the K-means method, such as [11, 54]. A drawback of some of these methods is that the number of clusters must be predetermined, which implies that a user should know in advance the region number of the image to segment. In order to overcome these limitations, methods based on representing an image as a graph have been introduced. One of the earliest graph-based methods dates back over 40 years and it is based on the minimum spanning tree (MST) of a graph [79]. This method gives a weight to edges on the basis of differences between pixel intensities, and breaks large edges by fixing a threshold. Improvements on the policy of edge breaking are proposed in [70]. Wu and Leahy [76] present a method based on the minimization of the concept of cut, which is the weight of edges connecting two regions. To avoid unnatural cuts of small groups of isolated nodes, in [65] a new measure of dissimilarity between two groups named *normalized cut* is introduced. More recently, Felzenszwalb and Huttenlocher [19] define a measure of evidence of the boundary between two regions by considering both the differences of intensity across the boundary and among neighboring pixels within a region.

2.4.1 Image segmentation for multispectral satellite images

Looking at the multispectral earth observation images generated from radars or scanners at multiple bands, there is a huge literature about segmentation techniques that can be contained into well-defined categories.
Histogram thresholding represents spectral variations using the image histogram: an appropriate threshold value is found on it to separate objects from the background [61]. Threshold can be obtained using some specific criterion such as maximization of class or entropy [52]. The main advantage of this model is the easiness and the efficiency of usage for image segmentation, because it considers only two levels. However, it is not able to represent the high histogram variability because of the threshold value [17]; in this sense, it cannot be used successfully for complex multispectral images. Furthermore, proximity information of pixels is not preserved and the histogram describes only global information.
Other approaches are based on the concept of **Markov Random Fields**. MRF is a model that considers the neighbourhood dependencies almost pixels [17]. In [7] a modified model considering multi-scale resolution is presented. It uses structures of quadtree and pyramid graph for scale representation and the expectation maximization algorithm to provide the parameters values. In some applications [62][78], an approach in two steps is performed on over-segmented images. A region adjacency graph for a MRF based on intra-region homogeneity and inter-region dissimilarity was developed. Then, region merging was performed based on the energy model. MRF is a very appealing approach for its ability to integrate spectral, textural and contextual information, although the implementation of the model is very complex and it is not able to represent the properties of shape and size.
Unsupervised neural networks such as Self-organizing Maps (SOM) have also been used for multispectral image segmentation. In [71] texture features based on co-occurrence matrix are given as input to a SOM for cloud detection. A Pulse coupled network characterized by the correspondence one to one to image pixels and the usage of neighbourhood relationship are introduced in [39], while an improvement of this technique in the sense of linking of neurons and reduced complexity is formulated in [46]. PCNN can be considered as a very promising approach to perform segmentation, because of the neighbourhood relationship inclusion. However, the decision of network structure is critical and some problems are related to learning and generalization of the network.
Watershed approaches are the edge detection algorithms that are currently used for multispectral image segmentation. The watershed model transforms the image into the corresponding gradient, so that it is considered as a structure where grey values correspond to elevation values at that position. The flood of water is spread from the minimum elevation values and when convergence is reached, a boundary is built across them [9].

In [12] a different gradient operator which envelopes textural information is presented. Furthermore, other marker-controlled watershed techniques are introduced in [6]. The main problem of watershed algorithm is the over-segmentation that it produces because of noise or textured patterns. For this reason, it can be used successfully as an initial segmentation approach in a multi-scale resolution technique or associated with a region merging algorithm [12] [14].
Multi-resolution techniques are the most frequently used approaches to segment multispectral images. They are based on the importance of factor of scale to identify the objects [4]. These approaches can be defined as top-down or bottom-up. In the first one, multispectral image is segmented at coarse level. Segmentation is the input for the following step that performs a finer partitioning. In the second one, the segmentation process goes in the reverse order from finer to coarser [81]. In [48] a fuzzy approach to select automatically the segmentation parameters used for multi-resolution is presented. Furthermore, a multiscale segmentation using region merging approach for initial segmentation and minimum heterogeneity rule for merging objects is introduced in [45]. The strong advantage of multi-resolution lies in the possibility to incorporate spectral, shape, size, textural and contextual information of regions at various scales for segmentation. Although this represents a very promising feature, scale and parameters of scale representation and information extraction from each scale remain a critical problem.
Fuzzy logic has been introduced into more traditional approaches to perform multispectral image segmentation adding fuzzy boundaries for objects. In [69] an iterative fuzzy clustering that uses spectral, spatial, textural and frequency information in fuzzy form is described. In [53] fuzzy techniques associated with histogram thresholding using fuzzy entropy, correlation and geometry are presented. This model is able to solve the ambiguity in segmentation using fuzzy logic. However, most of the fuzzy segmentation methods are derived from clustering and histogram thresholding [64], so the same problems mentioned before about histogram thresholding can be discussed. About clustering, as the image is mapped in the feature space, proximity information can be lost and regions obtained from segmentation algorithm can appear unconnected.

2.4.2 Evolutionary computation for image segmentation

In the last years much effort has been done in the definition of effective evolutionary-based approaches for solving complex problems of computer vision. In particular, evolutionary techniques have been successfully applied to the image segmentation problem, that we know to be of prior importance for facing more complex higher level problems such as *Object Recognition*. A survey on the application of genetic algorithms for image enhancement and segmentation can be found in [56]. Many of the approaches use a representation of the image based either on the cluster centers or on the label of the cluster a pixel is assigned to. A color image segmentation algorithm based on evolutionary approach has been proposed by Halder and Pathak in [30]. Each individual is a sequence of cluster centers and the cost function is the inverse of the sum of Euclidean distances of each point from their respective cluster centers. In order to determine the most appropriate number k of clusters, the algorithm is repeated for values of k equals to 2 until k_{max}. The choice of the best k is done by computing a cluster validity index based on inter and intra distances between clusters, while the value of k_{max} must be given as input to the algorithm. Jiao in [37] proposed an evolutionary image texture classification algorithm where the individuals are the cluster representatives, and the total distance between the pixels and the corresponding centroids is optimized. The distance for a couple of pixels is the value of the shortest path between them in the undirected weighted graph representing the image. In the same paper the author defines a Memetic Image Segmentation approach, where a genetic algorithm is applied on a set of regions previously extracted from a watershed segmentation in order to refine or merge the partitions into clusters. In this case each gene of a chromosome is the cluster label of the corresponding pixel. The association of the regions with the clusters is evolved by optimizing the total distance between the pixels and the corresponding centroids. In the former approach the number of clusters must be fixed a priori, while in the latter an approximate initial number is obtained by using the watershed segmentation, and then a final local search procedure merges regions to obtain the optimal number of clusters. Lai and Chang [40] proposed a hierarchical structure of the chromosome, composed by control genes, representing a partitioning in regions, and parametric genes, containing the representative gray levels of each region. The goal is to optimize a fitness function that is the sum of the distances between the gray level of each pixel and the representative gray level of its region. The number of control genes, as stated by the authors, is a $soft$ estimate of the upper bound of the number of regions. Merzougui et al. [49] proposed an evolutionary based image segmentation technique where the individuals are the components of the cluster centers and the fitness is the mean distance between the pixels and the centroids. In order to determine the optimal number of clusters, a criterion based on separability and compactness of the clusters is first applied. Di Gesú and Bosco [24] introduced an image segmentation algorithm where each chromosome represents the position and the region label where the pixel is located. The fitness function is defined on the similarity value and the spatial proximity between a pixel (chromosome) and the mean gray value of its corresponding region.

Because of the adopted representation, one of the main problems of the just described evolutionary approaches is the determination of the number of regions. Though different criteria are used to fix this number beforehand, the genetic algorithm cannot change this number while executing.

2.4.3 Our contribution

The segmentation method we propose in the following is genetic-based and it is called *Genetic Normalized Cut*. To the best of our knowledge, we introduce for the first time a genetic approach extending the Normalized Cut criterion. In fact, it dynamically computes the number of regions optimizing the fitness function that is an extension of the Normalized Cut concept of Shi and Malik [65]. However, differently from the traditional $Ncut$, we allow for a simultaneous k-way partitioning of the image without the need of fixing the number k of divisions beforehand. In fact, the adopted locus-based representation of individuals lets us to automatically determine the optimal value k of the objective function.

Similarly to [65], our approach uses a representation of the image as a weighted undirected graph. Each node is a pixel and the weights on the graph edges determine the distance between the pixels in terms of intensity, color or texture. However, in the $Ncut$ approach, given a pixel i, its neighbors are all those pixels which are no more than r pixels apart from i. We extend this concept by taking into account for the neighbourhood not only the spatial closeness, but also the pixel affinity. Consequently, neighbors of i will be those pixels which are "not too far" from i and that have also maximum similarity with i in terms of intensity, color or texture. It lets to improve the convergence speed of the method and the quality of the results.

2.5 *Genetic Normalized Cut* for image segmentation

In this section we give an informal description of the *Genetic Normalized Cut* algorithm and the variation operators adopted.

As previously mentioned, given an image I, it is represented as a weighted undirected graph $G = (V, E, w)$, whose nodes are the pixels and the edge weights correspond to affinity values among them. This means that the weight value associated with the edge linking two pixels will depend on the similarity between them in terms of intensity, color or texture. It's important to observe that two nodes of the graph will not have an edge between them if they are too far spatially or they have a low affinity value. It is mainly because in image segmentation the space constraint is a necessary but not sufficient condition to ensure that two pixels belong to the same region. It's important instead to pursue a trade-off between the spatial proximity and the similarity.

Given these preliminaries, we introduce $GeNcut$ as a genetic algorithm performing the following steps.

1. Create an initial population of random individuals each representing a segmentation solution.
2. While termination condition is not satisfied, perform the following sub-steps
   (a) evaluate the fitness of each individual
   (b) create a new population of individuals by applying the variation operators

The algorithm begins to generate a population initialized at random with individuals each representing a partition in sub-graphs of the image graph G. Consequently, an individual is a possible solution for the segmentation problem, because each sub-graph corresponds to a possible region of the segmented image.

The population is "repaired" to produce "safe" individuals, that is individuals containing connected sub-graphs of G. This is realized by linking to each node one of its neighbors. The neighbors of a pixel i are spatially not distant and similar to i in terms of intensity, color or texture. Consequently, an individual generating this kind of partitioning avoids uninteresting divisions containing pixels which are spatially too far or not quite similar to each other. This process is very useful because it improves the convergence of the method: in fact, the space of the possible solutions is restricted.

After that, the fitness function must be evaluated on the individuals of the population. Because it is an extension of the Normalized Cut concept, it measures both the dissimilarity between the different connected groups and the rest of the graph as well as the total similarity within the groups. Low values of our fitness function should be preferred: they let us to obtain a partitioning with strongly connected groups which are maximally dissimilar to the rest of the graph. In terms of image segmentation, this means to pursue a partitioning of the image in regions which are dissimilar each to the rest of the image but uniform inside in terms of intensity, color or texture. But it is exactly the goal that we have to reach if we consider the image segmentation problem.

About the variation operators, we use the *uniform crossover*: the child at each position i contains a value j coming randomly from one of the two parents. Intuitively, it corresponds to generate a new segmentation of the image from the two parent segmentations. The interesting property of this operator is that it is able to generate "safe" children from two "safe" individuals. It guarantees to moving in the search space through the only solutions which are connected sub-graphs, that is "interesting" partitions of the image, as previously defined.

Another variation operator is the mutation: it randomly modifies the value j of a i-th gene with one of its neighbors, in order to guarantee the generation of a "safe" mutated child in which each node is linked only with one of its neighbors. This means that the generation of the children segmentations will not diverge from the subspace of the good segmentation solutions: a neighbor is in fact a pixel with high affinity in spatial distance and similarity.

2.6 *GeNcut*: the methodology

Next, we formalize the concepts intuitively discussed in the previous section. First of all, the representation of the image as a weighted undirected graph is described. Then, the concept of normalized cut and the fitness function used by *GeNCut* are introduced. Finally, the genetic representation and employed operators are explained.

2.6.1 Problem definition

An image R can be represented as a weighted undirected graph $G = (V, E, w)$, where V is the set of the nodes, E is the set of edges in the graph, and $w : E \rightarrow \mathcal{R}$ is a function that assigns a value to graph edges. Each node corresponds to a pixel in the image, and a graph edge (i, j) connects two pixels i and j, provided that these two pixels satisfy some property suitably defined that takes into account both pixel characteristics and spatial distance. The weight $w(i, j)$ associated with a graph edge (i, j) represents the likelihood that pixels i and j belong to the same image region and provides a similarity value between i and j. The higher the value of $w(i, j)$, the more likely the two pixels are members of the same region. Let W be the adjacency weight matrix of the graph G. Thus W_{ij} contains the weight $w(i, j)$ if the nodes i and j are connected, zero otherwise. Depending on the method adopted to compute the weights, any two pixels may or may not be connected. In our approach we employed the *Intervening Contour* method described in [44, 16]. In this framework, given a generic pixel, the magnitude of the orientation energy at that pixel is considered. If the maximum image edge magnitude along a straight line connecting the two pixels i and j in the image plan is large, then a deep change and, consequently, an intervening contour is present, indicating that the two pixels do not belong to the same segment. Hence, the weight $w(i, j)$ between these pixels will be low. On the other hand, if the image edge magnitude is sufficiently weak, this usually happens in a region that is flat in brightness, the affinity between the two pixels will be very high. More formally, the weight $w(i, j)$ between the pixels i and j is computed as:

$$w(i,j) = \begin{cases} e^{-max_{x\in line(i,j)}||Edge(x)||^2/2a^2} & \text{if } ||X(i) - X(j)||_2 < r, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

where $a = (max_{y \in I}||Edge(y)||) \times \sigma$, *Edge(x)* is the image edge strength at position *x*, *I* is the image plan, *line(i,j)* is a straight line between i and j, *X(i)* is the spatial location of the pixel i, *r* is a distance threshold and σ is the image edge variance. In order to compute the weight between the pixels i and j, image edges across various scales are considered.

2.6.2 Objective Function

In the last few years many different criteria have been defined to partition a graph representing an image into non-overlapping connected components. Shi and Malik [65] introduced the dissimilarity measure *normalized cut* to divide a graph into two subgraphs, that revealed successful for image segmentation. The concept of normalized cut is an extension of the notion of *cut* proposed by Wu and Leahy [76] that avoids the bias for partitioning in small sets of nodes. Given a partition of a graph G in two disjoint sets of nodes A and B, the cut between A and B is defined as

$$cut(A,B) = \sum_{i \in A, j \in B} w(i,j)$$

In [65] the authors pointed out that the cut value diminishes when small sets of isolated nodes are generated. Thus a disassociation measure, that takes into account the total edge weight connecting two partitions, has been introduced. Let

$$assoc(A,V) = \sum_{i \in A, t \in V} w(i,t)$$

be the total connection from nodes in A to all the nodes in V, then the normalized cut is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

Shi and Malik formalize the problem of minimizing the normalized cut as a generalized eigenvalue problem and compute an optimal partition by using the eigenvector with the second smallest eigenvalue. Two extensions of the approach to k-way partitioning are also proposed. The former recursively partitions the groups obtained in the previous step by checking the values of the eigenvectors, the latter exploits the top n eigenvectors and the clustering algorithm K-means. A main limitation of this method is that the number k of desired partitions must be fixed beforehand.

We now introduce an extension of the concept of normalized cut that is used as criterion to partition a graph in a generic number k of regions. Note that, the value of k in our approach must not be fixed in advance, but it is determined by the optimal value of the objective function. Let $G = (V, E, w)$ be the graph representing an image, W its adjacency matrix, and $P = \{S_1, \ldots, S_k\}$ a partition of G in k clusters.

For a generic cluster $S \in P$, let

$$c_s = \textstyle\sum_{i \in S, j \notin S} W_{ij} \qquad m_s = \sum_{i \in S, j \in S} W_{ij} \qquad m = \sum_{i \in V, j \in V} W_{ij}$$

be respectively the sum of weights of edges on the boundary of S, the sum of weights of edges inside S, and the total graph weight sum. The *weighted normalized cut WNCut* measures for each cluster $S \in P$ the fraction of total edge weight connections to all the nodes in the graph

$$WNCut = \sum_{s=1}^{k} \frac{c_s}{m_s + c_s} + \frac{c_s}{(m - m_s) + c_s}$$

Note that c_s corresponds to $cut(A,B)$ where $B = V - A$. Because of the affinity measure w defined in the previous section, and the relationship between cut and $assoc$ formalized in [65], more uniform regions can be obtained with low cut values between the subgraphs representing the regions and the rest of the graph. This implies that low values of $WNcut$ should be preferred.

2.6.3 Genetic representation and operators

The genetic algorithm uses the locus-based adjacency representation proposed in [55]. In this graph-based representation an individual of the population consists of N genes $g_1, \dots, g_N$ and each gene can assume allele values j in the range $\{1, \dots, N\}$. Genes and alleles represent nodes of the graph $G = (V, E, w)$ modelling an image, and a value j assigned to the ith gene is interpreted as a link between the pixels i and j. This means that in the clustering solution found i and j will belong to the same region.

The initialization process assigns to each node i one of its neighbors j. This guarantees a division of the graph in connected groups of nodes. The kind of crossover operator adopted is uniform crossover. Given two parents, a random binary vector is created. Uniform crossover then selects the genes where the vector is a 0 from the first parent, and the genes where the vector is a 1 from the second parent, and combines the genes to form the child. The mutation operator, analogously to the initialization process, randomly assigns to each node i one of its neighbors.

The genetic operators need to determine the neighbors of each node. In our approach we introduced the concept of neighbors of a node by taking into account not only the spatial closeness, but also the pixel affinity. More in details, given a generic node i in the graph, let $w^h_{max} = \{w^1, \dots, w^h \mid w^1 \geq, \dots, \geq w^h\}$ be the first h highest weights of row i in the weight adjacency matrix W.

The h nearest neighbors of i, denoted as nn^h_i, are then defined as $nn^h_i = \{j \mid w(i, j) \in w^h_{max}\}$.

nn^h_i is thus the set of those pixels that are no more than r pixels apart from i, and that have maximum similarity with i. It is worth to note that, even if h is fixed to 1, the number of nearest neighbors of i could be sufficiently large if many of its spatial neighbors have the same maximum weight w^h_{max}. This definition of nearest neighbors guarantees to choose the most similar neighbors during the initialization process, and to bias the effects of the mutation operator towards the most similar neighbors, thus it contributes to improve the results of the method.

2.7 Results and discussion

In this section we present the results of *GeNCut* on five images with details of increasing complexity, and compare the performances of our algorithm in partitioning natural and human scenes in meaningful objects with the segmentations obtained by the algorithm of Shi and Malik [65] (in the following referred as $NCut$) on the same images. The *GeNCut* algorithm has been written in MATLAB 7.14 R2012a, using the Genetic Algorithms and Direct Search Toolbox 2. In order to set parameter values, a trial and error procedure has been employed and then the parameter values giving good results for the benchmark images have been selected. Thus we set crossover rate to 0.9, mutation rate to 0.2, elite reproduction 10% of the population size, roulette selection function. The population size was 100, the number of generations 50. The value h of nearest neighbors to consider has been fixed to either 1 or 2. As already pointed out, this does not mean that the number of neighbors is 1 or 2, but that the first (and second) most similar neighbors are taken into account for the initialization and mutation operators. The fitness function, however, is computed on the overall weight matrix. For all the data sets, the statistical significance of the results produced by *GeNCut* has been checked by performing a t-test at the 5% significance level. The p-values returned are very small, thus the significance level is very high since the probability that a segmentation computed by *GeNCut* could be obtained by chance is very low. The version of the $Ncut$ software that we used is written in MATLAB and it is available at http://www.cis.upenn.edu/ jshi/software/. The weight matrix of each image is the same for both methods, and, as already described in section 2.6.1, it is based on the Intervening Contour framework by fixing $r = 10$, number of scales 3, number of orientations 4 and $\sigma = 0.1$. Since $NCut$ needs the number k of clusters, we executed the algorithm by using two different inputs. The first sets the number k of segments to the same number of clusters found by *GeNCut*, the second one is a higher value. In the following, for each image, we compare the segmentation results of *GeNCut* and $NCut$ by depicting the contours of the regions obtained by the two approaches. For a more clear visualization, we show two images. The first image reports the boundary lines of the segmentation obtained on the original color image, the second one delineates the same contours without the image background.

Fig. 2.1 shows the execution of *GeNCut* and $NCut$ on an image of a melanoma. In particular, *Fig.* 2.1(a) is the original image, *Fig.* 2.1(b) and *Fig.* 2.1(c) display the segmentation obtained by *GeNCut* with and without the background image resp., while *Fig.* 2.1(d) and *Fig.* 2.1(e) are the results of $NCut$ when the number of segments is fixed to two, and *Fig.* 2.6(a) when this number is 5. *Fig.* 2.1(b)(c) show that *GeNCut* is able to find the right partitioning and correctly discriminates between the melanoma and the skin, although we do not set the number of segments a priori. *Fig.* 2.1(d)(e) and *Fig.* 2.6(a) point out that if $NCut$ receives the true number of segments, it is able to find the correct partitioning, otherwise an over-segmentation is obtained. In general, however, given an input image, it is hard to know the true number of partitions a priori.

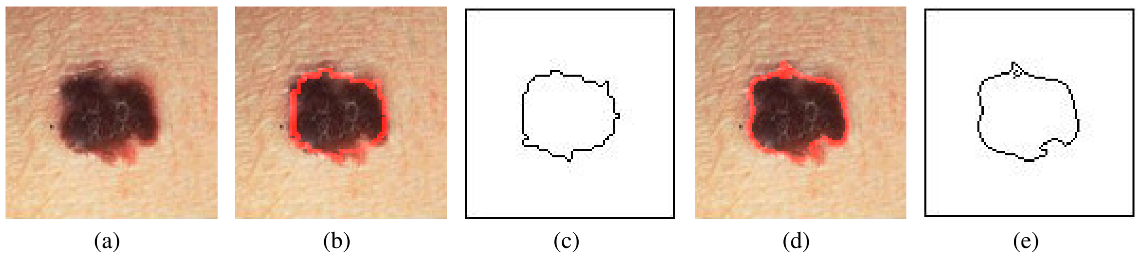


Fig. 2.1. (a) The original image representing a melanoma, (b) the segmentation result on the original image of *GeNCut* with h=2, (d) $NCut$ with $k = 2$ and (c-e) the corresponding contours.

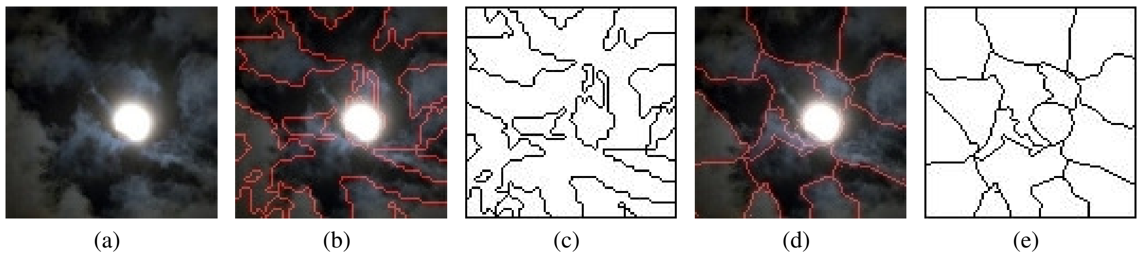


Fig. 2.2. (a) The original moon image, (b) the segmentation results on the original image using *GeNCut* with h=1, (d) $NCut$ with $k = 16$ and (c-e) the corresponding contours.

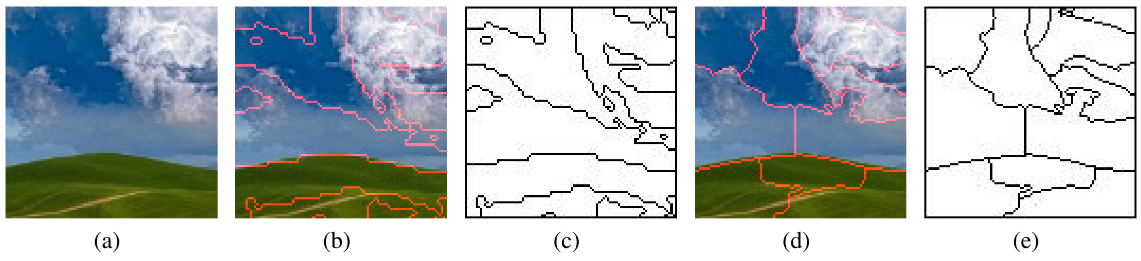


Fig. 2.3. (a) The original image, (b) the segmentation results on the original image using *GeNCut* with h=1, (d) $NCut$ with $k = 12$ and (c-e) the corresponding contours.

The next experiment represents a more complex scenario, due to the presence of irregular shapes (clouds) around a spherical object (moon) (*Fig.* 2.2(a)). *Fig.* 2.2(b)(c) illustrate the results obtained by using the *GeNCut* approach, while *Fig.* 2.2(d)(e) and *Fig.* 2.6(b) describe the outputs of the $NCut$ method when the number of segments is set to 16 and 22, respectively. Although the halo makes it difficult to segment the moon, by using our algorithm we are able to perform a segmentation that is more flexible in capturing the real shape of the clouds. $NCut$, instead, realizes a flatter partitioning with an equal number of segments that is not able to distinguish and capture some inner parts of the original image. *Fig.* 2.3(a) and *Fig.* 2.4(a) show two different

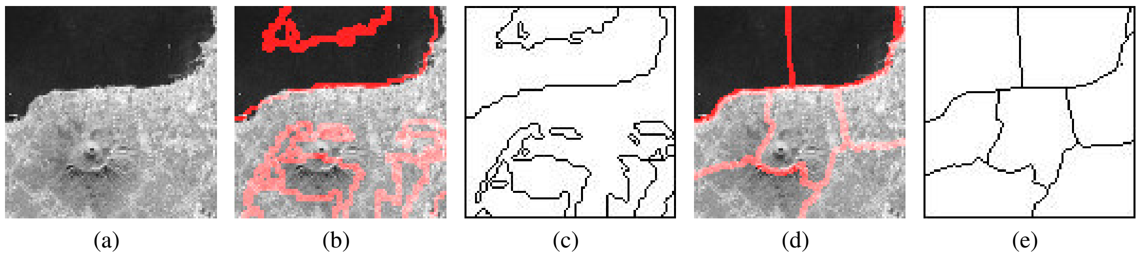


Fig. 2.4. (a) An X-SAR image of the Vesuvius volcano, (b) the segmentation results on the original image using *GeNCut* with h=1, (d) $NCut$ with $k = 8$, and (c-e) the corresponding contours.

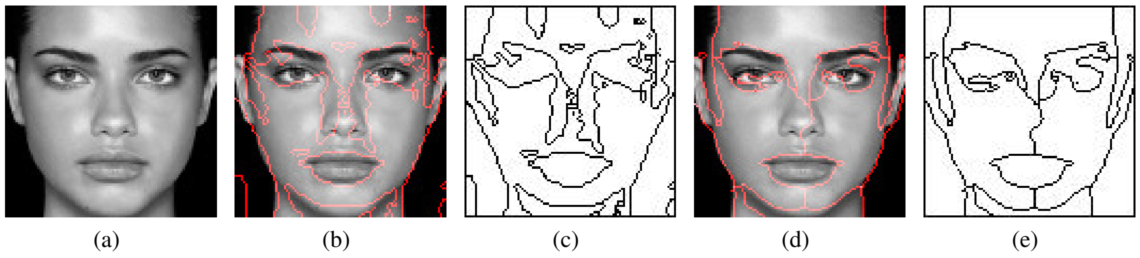


Fig. 2.5. (a) The original face image, (b) the segmentation results on the original image using *GeNCut* with h=2, (d) $NCut$ with $k = 12$ and (c-e) the corresponding contours.

kinds of landscapes: a natural picture and a snatch from an X-SAR image of the Vesuvius volcano (Italy), acquired by the Spaceborne Imaging Radar-C/X-Band Synthetic Aperture Radar (SIR- C/X-SAR) aboard the Space Shuttle Endeavour in 1994. For both the images, our algorithm is able to discover the meaningful objects, *Fig.* 2.3(b)(c) and *Fig.* 2.4(b)(c), respectively, while a poor segmentation of the major components like in *Fig.* 2.3(d)(e) and *Fig.* 2.6(c) is obtained by $Ncut$, despite the setting of the same number of segments, naturally extracted from our technique. The satellite image in *Fig.* 2.4(a) is a scene where it is quite difficult to differentiate the meaningful objects due to the details of the terrain. However, as it can be observed in *Fig.* 2.4(b)(c), *GeNCut* is able to separate the volcano area from the landscape and to distinguish the building barely visible at the bottom right corner and the area of the deep sea. All these significative features are not visible in the segmentation results of the $NCut$ approach, *Fig.* 2.4(d)(e), even if we increase the number of partitions, *Fig.* 2.6(d). Finally, we used *GeNCut* to segment a human face image (*Fig.* 2.5(a)). In this case the two approaches are comparable. Although more details are discovered from $NCut$ in correspondence of the eyes, it over-segments the face (*Fig.* 2.5(d)(e) and *Fig.* 2.6(e)). On the other hand, *GeNCut* obtains a uniform and natural segmentation of the face (*Fig.* 2.5(b)(c)) that is able to capture also the shape of the nose, although it appears linked to the eyes, probably due to the similar gray intensities along the contours of the nose and the contours of the eyes.

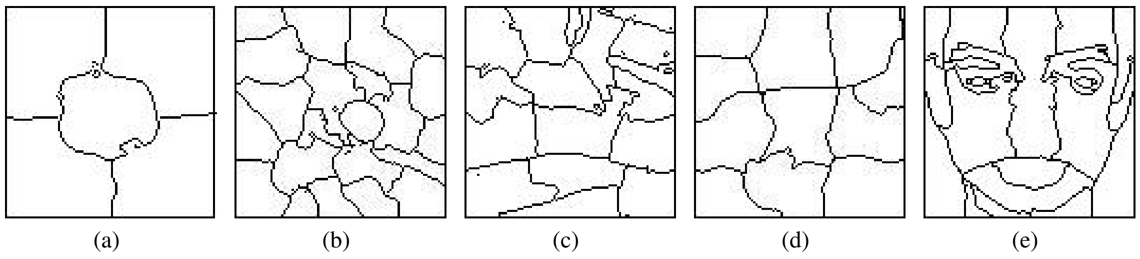


Fig. 2.6. The segmentation results representing the contours using $NCut$ with (a) $k = 5$, (b) $k = 22$, (c) $k = 20$, (d) $k = 12$, (e) $k = 20$.

3

Image Compression by 2D Motif Basis

3.1 Introduction

In this chapter, we study image compression techniques based on patches that are not necessarily solid blocks, but are affected instead by a controlled number of undetermined or *don't care* values. Such patches are chosen from a special set of candidate *motifs* that are extracted in turn from the image *2D motif basis*, the latter representing a particularly compact set of patterns resulting from the autocorrelation of the image with itself. As is expected, it is found that limited indeterminacy can be traded for higher compression at the expense of negligible loss.

Specifically, our encoding is based on a dictionary of patches consisting each of a suitable 2D motif possibly containing some unspecified or *don't cares* pixels. Whereas the total number of such patches may scale exponentially with the number of don't cares, it is possible to identify a subset of patches that is only linear in the size of the image and yet contains all the information about the entire repertoire of patches. Our proposed scheme consists of the iterated selection of the patch in this reduced set that conveys the highest estimated gain in compression. The scheme is thus offline and may be considered an extension to 2D of similar onedimensional schemata previously presented in, e.g., [1].

The chapter is organized as follows. The next section introduces the image compression problem. Then, section 3.3 describes the state of the art in image compression together with our personal contribution. After that, section 3.4 gives a brief overview of our technique. Section 3.5 recaptures from [2, 59] basic notions about motifs with don't cares and 2D motif bases and formally defines algorithms and implementations. Section 3.6 displays preliminary experimental results.

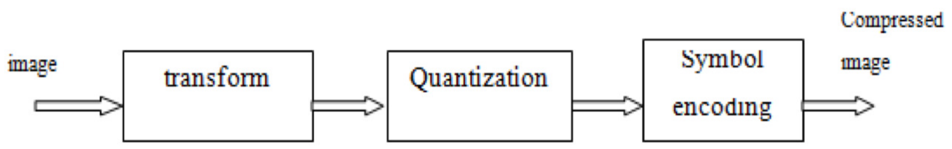
3.2 The image compression problem

Image compression consists in applying data compression on digital images. The main goal of this task is to reduce redundancy of the image data for storing or transmitting data as efficiently as possible. If graphics, audio and video are not compressed, they need a huge storage capacity and transmission bandwidth. Although the development of the modern technology (increase in mass-storage density, high speed processors, and more powerful communication systems), a strong request for data storage capacity and data transmission bandwidth is still present. In order to solve this problem, expecially in multimedia-based web applications where a massive usage of resources is required, the application of data compression to video and audio signals for storage and communication technology has been adopted.

A way of classifying compression techniques is by distinguishing between *lossless* and *lossy* compression. Lossless compression procedure does not introduce data loss during the compression phase. Consequently, the reconstructed image, after compression, is numerically identical to the original image. For this reason, lossless compression is not as powerful as the lossy one. Applying a lossy compression procedure introduces a loss on the compressed signal with respect to the original one. So, the image reconstructed from lossy compression is not fully identical to the original image, because redundant information is completely eliminated. For this reason, lossy compression is able to obtain much higher compression rates than the lossless one.

A lossy image compression system is typically composed of three closely connected components called (a) Source Encoder (b) Quantizer, and (c) Entropy Encoder. Compression procedure applies, first of all, a linear transform to decorrelate the image data. Then, a quantization of the resulting transform coefficients is performed. Finally, entropy codifies the quantized values. Next, each component is described more in detail.
Source Encoder (or Linear Transformer). Many linear transforms can be used which include Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT).
Quantizer. The main goal of a quantizer is to reduce the number of bits needed to represent the transformed coefficients by reducing the precision of those values. Since many values can be associated with a single quantization level, it is a lossy phase and the main source of compression in an encoder. Quantization on each individual coefficient is called Scalar Quantization (SQ). Quantization on a group of coefficients together, is called Vector Quantization (VQ). Uniform and nonuniform quantizers can be used depending on the context.
Entropy Encoder. An entropy encoder compresses the quantized values without loss to increase further the compression rate. A model is used to compute the probabilities for each quantized value. Then, a code based on these probabilities is produced in order to reduce the size of the input code stream. Generally, the Huffman encoder and the arithmetic encoder are the most commonly used entropy encoders, together with the run-length encoding (RLE) useful for applications requiring fast execution.

In the next section, lossless and lossy techniques are described in more detail and the main approaches constituting the state of the art in image compression are reported.



General Steps in Image Compression

3.3 State of the art in image compression

Image compression is broadly classified into lossless and lossy techniques. The most important approaches belonging to these two categories will be described in the following.

3.3.1 Lossless Compression Techniques

Lossless compression is mainly used in medical imaging, because it preserves the quality of an image during the compression phase. The most important approaches included in this category are the *Run Length Encoding*, the *Huffman Encoding*, the *LZW Coding* and the *Area Coding* [72].

Run Length Encoding

This method counts the number of adjacent pixels with the same value inside the image. Each sequence found is substituted by the single value followed by the count of the value inside the sequence [27]. In particular, the repeating string, called a run, is usually encoded into two bytes. The first byte represents the number of characters in the run and is called the run count. An encoded run may contain 1 to 128 or 256 characters; the run count usually contains the number of characters minus one (a value in the range of 0 to 127 or 255). The second byte is the value of the character in the run, which is in the range of 0 to 255, and is called the run value. A number of variants of run-length encoding is presented for compression in 2D. Image data is normally run-length encoded in a sequential process where a 1D stream is considered, rather than a 2D array of data. In sequential mode, the encoding process starts at the upper left corner and proceeds from left to right across each row (the X axis) to the bottom right corner of the image. Alternative RLE schemes are also proposed that encode data down the length of the image (the Y axis) along the columns or even that scan the image along the diagonals in a zig-zag fashion.

Different RLE schemes have been introduced to encode images. They are usually divided into classes based on the kind of atomic elements that they consider in the encoding phase.

Bit-level RLE schemes. They encode at bit-level considering runs of multiple bits in a scan line. It is particularly suitable for monochrome (black and white) images. In this kind of scheme, the runs are codified as single-byte packets from one to 128 bits in length. The seven least significant bits correspond to the run count minus one, and the most significant bit embeds the value of the bit run, which is 0 or 1. A run longer than 128 pixels is divided across multiple RLE-encoded packets.

Byte-level RLE schemes. Byte-level schemes are suitable for compressing images that are stored as one byte per pixel (i.e. grayscale). They encode at byte-level, finding runs of identical byte values and ignoring individual bits along a scan line. In the byte-level RLE scheme, the runs of bytes are encoded into 2-byte packets. The first byte is the run count of 0 to 255, and the second byte contains the value of the byte run. Also, this scheme adapts to the possibility to store literal, unencoded runs of bytes within the encoded data stream. In particular, the seven least significant bits of the first byte are the run count minus one, and the most significant bit of the first byte indicates, if it is set to 1, indicates an encoded run. Encoded runs are decoded by reading the run value and repeating it the number of times indicated by the run count. If the most significant bit is 0, a literal run is expected: this run, whose number of repetitions is indicated by the run count, should be read literally from the image.

Pixel-level RLE schemes. It is used when a single pixel value is represented by two or more consecutive bytes inside the image (i.e. color 24 bit images). At this level, bits are ignored, and bytes are counted as parts of a pixel. The size of the encoded packet depends on the size of the pixel values being encoded. The number of bits or bytes per pixel is stored in the image file header. A run of an image composed of 3-byte pixel values is stored as a 4-byte packet: one run-count byte is followed by three run-value bytes, one for each channel inside the image (i.e., RGB).

Huffman Encoding

Huffman code is an effective and efficient technique for image compression [34]. It is a variable-length code: it encodes very frequent characters by using short codewords and characters which occur infrequently by using long codewords. Huffman code is also a prefix code. This means that there are no codewords which are prefix of other codewords. This is very important because ambiguity is avoided inside the codified data: codewords start differently of one another, in such way that the codewords inside the codified file are unambiguous. A binary tree is an efficient representation of a prefix code, whose leaves are the characters of the alphabet. The binary code of a codeword can be considered as the path along the tree from the root to the leaf representing the codified character.

The Huffman coding for digital image compression is based on entropy coding and realizes the coding idea discussed above [63]. For example, we usually need $n = 8$ bits to represent each pixel of a grayscale image, because the pixels of this kind of images can take $2^8 = 256$ possible brightness values. Furthermore, nM^2 bits are necessary to represent an image of $M \times M$ pixels. If the Huffman coding is used, however, it is possible to reduce the number of bits to codify each pixel (on average fewer than $n = 8$ bits).

Basically, Huffman coding represents a set of N characters of a digital image (e.g., $N = 256$ gray levels in a grayscale image) with binary code of variable length by the following steps to build the binary tree:

1. compute the probability of occurrence P_i for each of the symbols ($i = 1, 2...N$);
2. sort these probabilities in ascending order (from the smallest to the highest);
3. combine the two smallest P_i at the bottom and resort their sum with all other probabilities, repeat this phase until only two probabilities are left;
4. add a bit (0 or 1) to the binary codes of the two probabilities newly appearing at each step and repeat it until all the initial N symbols are encoded.

As output of the Huffman coding, all symbols are encoded optimally, such that more frequent symbols are encoded by shorter binary codes. Consequently, the average length (number of bits) of the codes for these symbols is minimized.

Dictionary Coders

This category includes the Lempel-Ziv algorithms [83, 84]. In particular, the Lempel-Ziv Welch is the base technique for GIF and often for TIFF image file formats [75]. Using this technique, the input image is linearly scanned row by row and the repetitions of recurring patterns in the input string are replaced by shorter references to a dictionary containing the original pattern. Coding can be static or dynamic: in the static one, the dictionary is fixed a priori for the encoding and decoding phases while in the dynamic mode, the dictionary is built in a single pass during data encoding. In this last mode, it is not necessary to transmit the dictionary: it can be recovered from the decoder while it decompresses the data.

An overview of the encoding algorithm is reported in the following:

1. Initialize the dictionary D to contain all the strings of the alphabet.
2. Search the longest string W in the dictionary matching the current input.
3. The dictionary index for W is the current output; then, remove W from the input.
4. Add W followed by the next symbol of the input to the dictionary D.
5. Repeat from step 2.

A dictionary D is created to contain the single-character strings which are all the possible characters of the input string alphabet. The algorithm proceeds by scanning step by step the input string by considering successively longer substrings until it finds one that is not in the dictionary. When such a string is found, the index associated with the string but the last character (corresponding to the longest substring in the dictionary) is removed from the dictionary and sent to output, and the new string (including the last character) is added to the dictionary with the next available code. After that, the last input character begins the next starting point to continue the scan. Step by step, increasingly long strings are stored inside the dictionary and made available for subsequent encoding as single output values. The algorithm is particularly suitable for data with repeated patterns. Consequently, the initial parts of a message will be slightly compressed.

Area Coding

This kind of coding consists in using some special codewords to represent large areas of contiguous 0s and 1s in order to compress a binary image [66]. The technique developes in the following steps:

1. divide the input image into blocks of size $m \times n$ pixels, where m and n are fixed a priori;
2. identify each block into the classes of all white, all black, and mixed intensity;
3. assign 1-bit codeword 0 to the class which occurs more frequently and assign 2-bit codewords 10 and 11 to the other two classes;
4. the code assigned to the mixed intensity class is just used as a prefix, which is followed by the $mn - bit$ pattern of the block.

Compression is obtained because large areas of equal bits are substituted with a single or double bit code, eliminating the redundant information. It is important to observe that area coding preserves the two dimensional character of images: they are not considered as a sequential stream but as a bidimensional object whose rectangular regions are used for compression.

3.3.2 Lossy Compression Techniques

Different kinds of lossy compression approaches are available in literature [72] and described in the following. The main discriminating feature of these methods is the quality of the reconstructed image. Furthermore, their application depends on their adequacy in specific contexts.

Vector Quantization

Goal of the vector quantization is to codify an input vector extracted from an image with a codeword based on a error measure criterion [15]. It is a pattern matching procedure: the index associated with the codeword is sent to the receiver which is able to recreate data by accessing to the codeword from that index.

In particular, the input image is divided into n-dimensional vectors. They can be patches of size $n = p \times q$ or a three dimensional vector composed of the pixel RGB values. Each vector extracted from the image is matched to a set of representative codevectors, $\hat{X}_i, i = 1...S$ taken from a previously created codebook. Codevector exhibiting the best match is selected by using a minimum distortion rule (distortion can be computed by using i.e. euclidean distance or mean absolute error). For example, given a vector X and the set of codewords $\hat{X}_i, i = 1...S$, $\hat{X}_z$ will be chosen, such that $d(X, \hat{X}_z) \leq d(X, \hat{X}_j)$ for all $j = 1...S$. Observe that $d(X, \hat{X})$ represents the distortion obtained in replacing the original vector X with the codevector $\hat{X}$.

After the codevector exhibiting the minimum distortion has been found, the index k of that codeword is transmitted to the receiver using $log_2 S$ bits. When the decoder receives the index, it uses a lookup table to find the codeword corresponding to that index. The found codeword is a duplicate of the original codeword and it is useful to reproduce the content of the image. Compression is obtained because the set of the codewords used to codify the image content is relatively small if compared to the original image vectors.

Fractal Coding

Fractal compression builds an approximation of the original image that is closer to it at higher compression ratios. Furthermore, compression result is quite accurate. [20].

In order to understand this kind of coding, it is important to discuss briefly about the *affine transformations*. An affine transformation of an image is any combination of a rotation, scaling, skew or translation. In matrix notation, a general affine transformation can be expressed as $x' = Ax + d$, where x is a vector of coordinates, A controls rotation, scaling and skew while d defines the translation.

A special kind of affine transformation is a *contractive transformation*. An affine transformation f is contractive if it moves points closer together. More formally, for all x and y, $d(f(x), f(y)) \leq sd(x, y)$, where s is a constant between 0 and 1 and $d(x, y)$ is the distance between the points x and y. A notable property of this transformation is that, taking a point and iterating the transformation many times to the successive results, the generated sequence of points moves closer and closer to a fixed point or limit z. This can be defined as $\lim_{n \to \infty} f_n(x) = z$. A contractive transformation can generate a fractal: by iterating the transformation an infinite number of times any image pattern is repeated many times with the same structure at any level of detail. Remember that this is one of the basic features of a fractal set.

Given a set of contractive transformations with parameters $c_1, c_2, ..., c_n$, an *Iterated Function System (IFS)* can be defined transforming each point inside the image by each transformation. In particular, it computes $c_1(x) \cup c_2(x) \cup ... \cup c_n(x)$. If A is a set of points, IFS works on A by applying each of the affine transformations and by calculating different images, one for each transformation. The final result of this computation is the union of the images. Again, if the IFS is iterated many times on the successive results, the output converges to a particular collection of points, called *attractor* of IFS, which is the fixed point of the transformation.

Consequently, given the input image to compress, the idea is based on finding an IFS that will generate the given image and storing the IFS coefficients in place of the original for performing compression. In order to find an IFS that will generate a given image, two properties should be considered:

1. *collage theorem*. If an IFS applied once to the input image gives an image that is very similar to the original one, then, the iteration of the IFS many times will give a result that is even more similar to the original image;
2. *continuity condition*. If the coefficients of the IFS are altered, then the changes on the attractor will be smooth and without any sharp jumps.

Consequently, the compression algorithm should change the control coefficients until it finds an image which is a suitable copy of the original image itself under a set of contractive affine transformations. This means to find an affine transformation that maps the whole image onto part of itself.

However, this theory about contractive transformations is still valid if such transformations are constrained to work only on parts of the whole image, mapping one part of the image onto another part of the image. An automatic approach divides the image into domain and range blocks, considering a smaller set of contractive affine transformations. The domain blocks are without overlaps and cover the entire image surface. The range blocks are bigger and partially overlapped: they contain parts of the image that will be mapped to the domain blocks by contractive affine transformations. For this reason, the compression algorithm consists on the following steps:

1. Divide the image into range blocks.
2. Divide the image into domain blocks.
3. For each domain block:
   a) compute the effect of each transformation on each of the range blocks;
   b) find the combination of transformation and range blocks obtaining the highest similarity to the image in the domain block;
   c) store the range block adopted and the transformation.

The output fractal compressed image is just a list of range block positions and transformations.

Block Truncation Coding

In block truncation coding, the input image of N pixels is divided into smaller patches of size m. Every patch is then processed separately [23], searching for a compact representation of that patch. In the decompression phase, the compressed patches are transformed back into pixel values so that the decompressed image is similar to the original one as much as possible.

For every patch, the mean value $\bar{x}$ and the standard deviation σ are computed as defined below:

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \bar{x^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \qquad \sigma = \sqrt{\bar{x^2} - \bar{x}^2}.$$

After that, a two-level quantization is performed. If a pixel exhibits a value which is less than the quantization threshold ($x_i < x_{th}$), it is codified to value a. If a pixel has a value which is greater than or equal to the threshold ($x_i \geq x_{th}$), it is codified to value b.
The mean value $\bar{x}$ is considered to be the threshold x_{th} and the values a and b are computed from the first and second sample moments ($\bar{x}, \bar{x^2}$), such that mean and standard deviation are preserved in the compression phase. Consequently, a and b are defined as:

$$a = \bar{x} - \sigma\sqrt{\frac{q}{m-q}} \qquad b = \bar{x} + \sigma\sqrt{\frac{m-q}{q}}$$

where q represents the number of pixels $x_i \geq x_{th}$.

A compressed patch is a triple $(\bar{x}, \sigma, B)$, where $\bar{x}$ and σ are the mean and standard deviation of the pixel values in the patch and B is the bit plane representing the quantization of the pixel values.

Sub Band Coding

The approach of Sub Band coding subdivides the source output into components based on frequency. In particular, in this compression approach used also in image processing, digital filters are used to divide the source output into different bands of frequencies. Then, each of these components will be encoded separately. The general sub band encoding procedure is described as follows:

1. Choose a set of filters for decomposing the source.
2. By these filters, construct the sub band signals.
3. Decimate the output of the filters.
4. Encode the decimated output.

First of all, the source output is separated into its constituent components by digital filters. It is realized by passing the source output through a bank of filters. This filter bank covers the range of frequencies the source output is composed of.

Remember that a filter is a system that selects only certain frequencies. Filters can be classified as:

- low pass filters;
- high pass filters;
- band pass filters.

The passband of each filter defines what kind of frequencies can pass through. Each of the components obtained from the filtering will be different bands of frequencies which constitute the source. The second step consists in subsampling of the outputs of the filters in order to reduce the number of samples. Finally, the decimated output is encoded by one of several encoding schemes, i.e. ADPCM, PCM, or vector quantization.

The inverse of the encoding procedure is the decoding procedure, where the quantized and coded coefficients are used at the decoder to reconstruct a representation of the original signal.

Transformation Coding

In this kind of coding, an input $M \times N$ image is divided into $p \times q$ sub-images or blocks. Each block is transformed and coded separately [5]. The transformation of each block of data is performed so that a huge part of its overall energy is embedded in relatively few transform coefficients. It corresponds to represent the image as the linear combination of some basis images and to specify the linear coefficients. Such coefficients are then quantized independently. DFT (Discrete Fourier Transform) and DCT (Discrete Cosine Transform) are transforms which are adopted to modify the pixels in the original image into transform coefficients.

Coefficients exhibit several properties. Two of such properties are:

- *Energy compaction.* Most of the energy of the original data is contained in only a few of the meaningful transform coefficients. This means that few basis images are sufficient to codify a given image.
- *Decorrelation.* Coefficients for separate basis images are uncorrelated.

Consequently, few coefficients which are particularly meaningful are then chosen and the remaining are eliminated. The selected coefficients are considered for further quantization and entropy encoding to produce the codewords.

3.3.3 The JPEG Standard

The most important algorithm for image compression is defined in the JPEG standard (JPEG, 1992) that was born in the 1980s and introduced by the Joint Photographic Experts Group (from which the name of the standard). In the last years it began the most used way of compressing pictures in various kinds of important applications, such as image transmission on the Internet and image storage for digital cameras. This algorithm is able to efficiently codify colour images in terms of average compression rate with good visual quality.

The JPEG standard works in four modes, three of them are lossy and the other one is lossless. All the lossy modes use the two-dimensional Discrete Cosine Transform (2D-DCT) to analyse the spatial-frequency features of the images in order to store with less precision (or even remove) those frequency components least important for a human observer (according to the human visual system). The DCT is also able to achieve high compactness of the information: after the application of the DCT, a substantial part of the image information is reduced in only a few low frequency transform coefficients and thus it can be efficiently represented.

The simplest and best known compression mode that is extensively used is the sequential one. In this mode, the input image can be a greyscale image or a colour image. A colour image can be represented not only in the RGB space but also it can be stored more efficiently in YCbCr space, in which the luminance (Y), the blue and the red chrominance (Cb and Cr respectively) are components which are represented separately to each other (the remaining green component, Cr, can be inferred from the Cb, Cr and Y components). The YCbCr colour representation allows us to reduce the size of the chrominance components without a significant degradation of the perceptual image quality, because the human eye is more sensitive to brightness information than to colour one.

Consequently, a colour space transform from the input colour space to YCbCr is considered as the first step of the JPEG algorithm. After that, a chrominance downsampling is performed, where the chrominance components are reduced by two in both horizontal and vertical directions, or only in the horizontal direction.

In the sequential mode, each image component is divided into 8×8 non-overlapping blocks, and they are compressed and transmitted (or stored) in scan order, from left to right, and from top to bottom, so that the decoder can recover the image step by step, in the same order of the encoding. Each block is then processed as follows:

1. On the entire block, the two-dimensional DCT is computed (it can be separately computed by using a 1D-DCT, first on the rows and then on the columns).

2. Then a quantization of the transform coefficients is performed to reduce information, most of all in high frequency components. This step introduces information loss in the encoding phase. In the quantization process each coefficient is divided by an associated constant value from a quantization matrix, rounding the result obtained to the nearest integer. This matrix is such that the higher frequency a coefficient represents, the higher the denominator (quantization value) becomes, and this corresponds to more information reduction.
3. The DC coefficient of the current block is differentially encoded by using as a reference the DC coefficient of the previous block.
4. The rest of the coefficients are scanned in zig-zag order, from lower to higher frequencies, and joint run-length and entropy coding are executed. With the run-length coding, the coefficient values and the count of zero-values are considered. Remember that compression is performed by eliminating an easy *spatial redundancy* associated with the sequences of identical pixels. Entropy coding is a statistical compression technique. It represents symbols by a bit sequence whose length is inversely proportional to the probability of its appearance (i.e., more likely symbols are encoded with fewer bits, and vice versa).

In addition, it is possible to vary the compression ratio by growing the value of elements in the quantization matrix. This process reduces image quality and it is called rate control.

Fig. 3.1 illustrates the JPEG compression process.

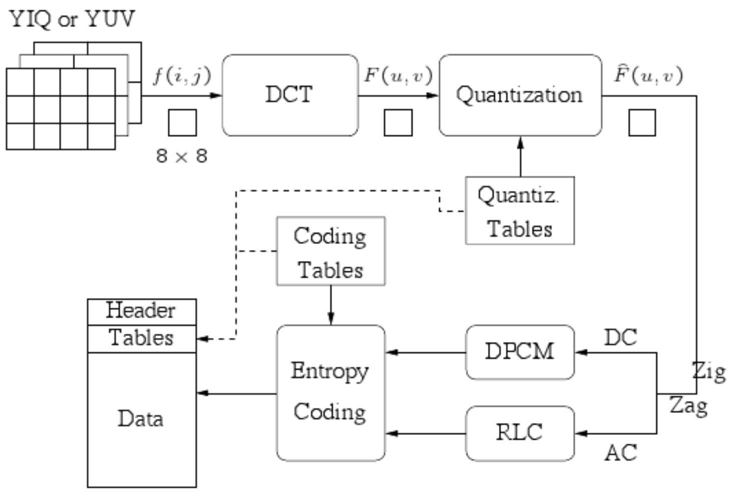


Fig. 3.1. Block diagram for JPEG encoder.

The other two lossy modes of JPEG are the progressive and the hierarchical ones. They are based both on sequential mode, but differently from it, each image component is not codified in a single scan.

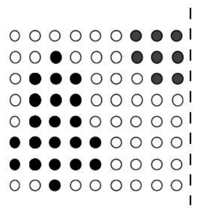
Progressive mode generates and transmits different versions of the image at increasing quality: first It sends an early version of low quality. Then, the details are added by successive scans (e.g. The DC coefficients and few AC coefficients are sent first. More AC coefficients are sent in succession until all coefficient sequence is complete.)

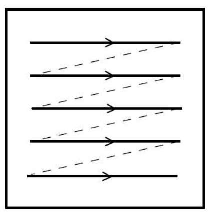
In the hierarchical mode, a first downsampling is performed on the original image by two several times. Then, an encoding and transmission as in the sequential mode is executed. After that, the encoder transmits the error committed by upsampling by two the low-resolution version of the image, and it repeats this process until the original resolution is obtained.

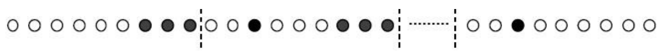
The lossless mode of JPEG has been introduced some years after the JPEG standard release. It is completely different in concept and usage from the lossy modes, in fact no block-division and transform is provided.

3.3.4 Our contribution

Let's recall that in last years, extensions to 2D of the classical LZ parses [42, 43, 74, 83, 84] have been proposed and used for image compression. Remember that, given an image (a), it is linearly scanned row by row (b) in order to obtain a linearized version for dictionary computation (c).


(a) An image


(b) Linear scan


(c) Linearized for dictionary computation

Although the technical implementation of 2D parses such as, e.g., relying on 2D subword trees [3, 25] may be quite demanding, the advantages of the 2D parsing over linear parsing are obvious. As an example, a linear parse of the image in Figure 3.2 would lead to extract a number of linear segments of mixed colors, but would miss the recurring large rectangular regions that cover almost the whole image.

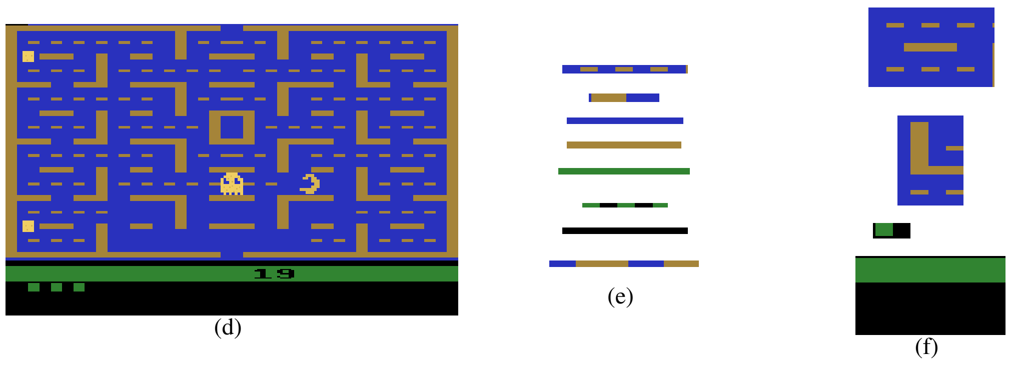


Fig. 3.2. Linear patches vs rectangular patches in the Pac-man.

Also, with most images, it is safe to omit a fraction of the pixels without compromising quality. The missing pixels may be approximated by interpolation at the decoder, often with negligible loss.

Consequently, our contribution was mainly focused on the introduction of a new image compression technique, based on the concept of *2D motif basis*, previously defined by A. Apostolico et al. in [2].

In our technique, parsing of the input image in rectangular patches is performed and not limited to the extraction of solid patches, that is full submatrices of characters from the image. In fact, our patches are composed of solid characters and *don't cares*, indicating that some pixels can be safely omitted from the decoder in decompression phase. Such patches are chosen from a set of maximal and irredundant motifs called $basis$ and selected to yield the highest estimated gain in compression. This lets us to obtain higher compression with negligible loss.

3.4 Image compression algorithm based on 2D Motifs

Next, we informally introduce the main steps of our image compression algorithm.

Basically, the technique extracts from an input image I a set of 2D non solid patches occurring in some points inside the image and which are able to encode the main image content. After that, a procedure is performed to substitute into the image the patches exhibiting the highest gain in compression with the pointers to these patches. The result of this procedure is a multiset pixels-pointers and a dictionary of patches sorted according to the gain.

More specifically, the compression procedure is composed of four main steps which are summarized in the following:

1. Compute a set of not-solid 2D patches from the input image I, called $basis$. Every patch occurs in some points inside I.
2. Compute a dictionary of patches from basis.
   a) For every patch, evaluate a measure of compression gain and store the patches into a dictionary sorted according to the gain.
   b) Eliminate all the occurrences of the patches overlapping with some other occurrences inside the image.
   c) If some patches don't occur anymore inside the image, eliminate them from the dictionary.
3. Associate a pointer to every patch of the dictionary.
4. For every patch, substitute inside the image each occurrence with the corresponding pointer to that patch.

The first step consists in finding a set of *maximal* and *irredundant* motifs from the input image, which are patches with *don't cares* whose number is linear in the size of the image. This set of motifs is called $basis$ and it is able to generate all the other motifs of I. The procedure to find the basis set can be summarized in the following steps:

1. Compute the set A of the autocorrelations of I.
2. For each autocorrelation in A, find the points where it occurs inside I.
3. Discard redundant motifs from A.
4. Remaining motifs are the basis set B.

The autocorrelation set A is computed by starting from the bottom-right corner of I and moving from the right to the left and from the bottom to the top. Each submatrix found along this path is matched to the image by positioning at its top-left corner: a pixel-mismatch between I and the submatrix at a given position is substituted by a *don't care*, while a relative match maintains the common pixel at that position. The same procedure is repeated by starting from the bottom-left corner of I and moving from the left to the right and from the bottom to the top. Each submatrix along this path is matched to the image by positioning at its top-right corner. The new motifs so obtained by scanning the whole image constitute the autocorrelation set A of I.

An intuitive example of the procedure to compute the basis set is reported below.
Given a binary image I:

$$I_{[4,3]} = \left\| \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix} \right\|$$

The autocorrelation A_{31} is obtained from I by matching the submatrix of I starting at position (3,1) with the image itself by positioning at its top-left corner. Columns or rows with only *don't cares* are eliminated from the autocorrelation.

The autocorrelation set depicted below is computed by performing the previously described procedure on the image I. Then, for each autocorrelation, the list of the points where it occurs inside I is found.

From the autocorrelation set A, we eliminate redundant motifs to compute the basis. For example, given the autocorrelation A_{23}, it is redundant because it is positioned exactly where are the other three autocorrelations A_{33}, A_{22}, A_{21} up to some offsets inside I and it is a submotif of the other three autocorrelations.

$$I_{[4,3]} = \begin{Vmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{Vmatrix} \qquad A_{31} = \begin{Vmatrix} 1 & 0 \\ 0 & \circ \end{Vmatrix}$$

autocorrelations: $A_{42}=\begin{Vmatrix} 1 & 0 \end{Vmatrix}$ $([1,1],[3,1],[4,2])$ $A_{41}=\begin{Vmatrix} 0 \end{Vmatrix}$ $([1,2],[1,3],[2,1],[2,2],[3,2],[4,1],[4,3])$

$A_{33}=\begin{Vmatrix} 1 \\ 0 \end{Vmatrix}$ $([1,1],[3,1],[3,3])$ $A_{31}=\begin{Vmatrix} 1 & 0 \\ 0 & \circ \end{Vmatrix}$ $([1,1],[3,1])$

$A_{23}=\begin{Vmatrix} 1 \end{Vmatrix}$ $([1,1],[2,3],[3,1],[3,3],[4,2])$ $A_{22}=\begin{Vmatrix} 0 & \circ \\ 1 & 0 \end{Vmatrix}$ $([2,1],[3,2])$

$A_{21}=\begin{Vmatrix} 0 & \circ \\ 0 & 1 \end{Vmatrix}$ $([1,2],[2,2])$ $A_{12}=\begin{Vmatrix} \circ & 0 \\ 0 & \circ \end{Vmatrix}$ $([1,1],[1,2],[3,1])$

Eliminating all the redundant motifs, the basis is obtained from I. After the basis extraction procedure, the dictionary of the patches must be computed from the basis. It will contain a subset of the motifs of B which exhibit the highest gain in compression and whose occurrences are not overlapped to each other inside I.

$$A_{23}=\begin{Vmatrix} 1 \end{Vmatrix} \quad ([1,1],[2,3],[3,1],[3,3],[4,2])$$

$A_{33}=\begin{Vmatrix} 1 \\ 0 \end{Vmatrix}$ $([1,1],[3,1],[3,3])$ $A_{22}=\begin{Vmatrix} 0 & \circ \\ 1 & 0 \end{Vmatrix}$ $([2,1],[3,2])$ $A_{21}=\begin{Vmatrix} 0 & \circ \\ 0 & 1 \end{Vmatrix}$ $([1,2],[2,2])$

Basis: $A_{33}=\begin{Vmatrix} 1 \\ 0 \end{Vmatrix}$ $([1,1],[3,1],[3,3])$ $A_{31}=\begin{Vmatrix} 1 & 0 \\ 0 & \circ \end{Vmatrix}$ $([1,1],[3,1])$

$A_{22}=\begin{Vmatrix} 0 & \circ \\ 1 & 0 \end{Vmatrix}$ $([2,1],[3,2])$ $A_{21}=\begin{Vmatrix} 0 & \circ \\ 0 & 1 \end{Vmatrix}$ $([1,2],[2,2])$ $A_{12}=\begin{Vmatrix} \circ & 0 \\ 0 & \circ \end{Vmatrix}$ $([1,1],[1,2],[3,1])$

The compression gain of a given motif determines how much saving I will obtain if I substitute the occurrences of the motif with a pointer to that motif inside I. It is function of the size and of the frequency of the motif inside the image.

For example, given the basis set $B = \{A_{33}, A_{31}, A_{22}, A_{21}, A_{12}\}$ previously computed, suppose to have the dictionary $\mathcal{D} = \{\mathbf{A_{12}}, \mathbf{A_{33}}\}$ extracted from the basis by considering the compression gain. The association of the motifs of the dictionary with the corresponding pointers and the substitution of the occurrences of each motif with the corresponding pointer are depicted below.

$$I_{[4,3]} = \begin{Vmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{Vmatrix} \qquad \mathcal{D} = \{< A_{12}, 1 >; < A_{33}, 2 >\} \qquad \begin{Vmatrix} 2 & 1 \\ 1 & 2 \end{Vmatrix}$$

Now, after the informal description of our technique, we give in Fig. 3.3 an appetizer of how the algorithm works on a real image.

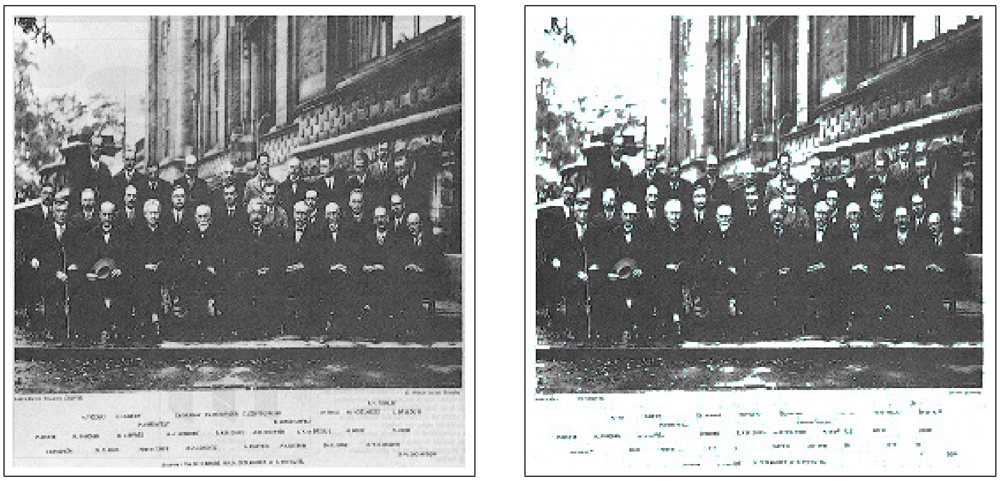


Fig. 3.3. Results of image compression by 2D Motif Basis on a real image. From left to right, the original image and the compressed image.

3.5 Image compression by 2D Motif Basis: the methodology

In the following section, we formally define the notion of $basis$. Then, we describe all the features of our image compression algorithm. Finally, we report some preliminary experimental results.

3.5.1 Basic notions

It is assumed that the input consists of a digitized *image* represented as a rectangular array of $N = m \times n$ of *pixels*, where each pixel i_{ij} is a *character* (typically, encoding an integer) over an alphabet Σ (see Figure 3.4 (a)). The notion of a suffix for a string translates into that of a *bite* in 2D. Figure 3.4 (b) shows the bite of I that starts at position (i, j).

$$I_{[m,n]} = \left\Vert \begin{matrix} i_{11} & i_{12} & \dots & i_{1n} \\ i_{21} & i_{22} & \dots & i_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ i_{m1} & i_{m2} & \dots & i_{mn} \end{matrix} \right\Vert \qquad S_{ij} = \left\Vert \begin{matrix} i_{ij} & i_{ij+1} & \dots & i_{in} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ i_{mj} & i_{mj+1} & \dots & i_{mn} \end{matrix} \right\Vert$$

Fig. 3.4. (a) An image I over an alphabet Σ. (b) The *bite* of I that starts at position (i, j)

In addition to the *solid* characters from Σ, we also deal with a special *don't care* character, denoted by '$\circ$', that is a wildcard matching any character in $\Sigma \cup \{\circ\}$. Some properties hold for characters:

1. σ_1 is a don't care character, then $\sigma_1 \prec \sigma_2$ (a *don't care* character is always more general than any other character in Σ).
2. If both σ_1 and σ_2 are identical characters in Σ, then $\sigma_1 = \sigma_2$.
3. If either $\sigma_1 \prec \sigma_2$ or $\sigma_1 = \sigma_2$ holds, then $\sigma_1 \preceq \sigma_2$.

An image P of size $m_p \times n_p$ on $\Sigma \cup \{\circ\}$, with $m_p \leq m$ and $n_p \leq n$, has an *occurrence* at position $[k, l]$ ($k \leq m - m_p + 1$ and $l \leq n - n_p + 1$) in I if $P[i, j] \preceq I[k + i - 1, l + j - 1]$ holds for $1 \leq i \leq m_p$ and $1 \leq j \leq n_p$. This means that P will occur in I from a given position $[k, l]$ if every character of P is the same or more general than each character in I by considering the same positions in P and I.

Given an image I on the alphabet Σ and a positive integer *quorum* $q \leq m \times n$, an image M on $\Sigma \cup \{\circ\}$ is a *q-motif* of I with *location list* $\mathcal{L}_M = (l_1, l_2, \dots, l_p)$, where each l_i is a pair of indices of I, if all of the following hold:

1. there is at least one solid character adjacent to each edge of M;
2. $p \geq q$;
3. M occurs at each $l_i \in \mathcal{L}_M$;
4. there is no location $l, l \neq l_i$, $1 \leq i \leq p$, such that M occurs on I (the location list has maximum size).

Basically, the image M is a *q-motif* in I if it occurs at least q times in I and if its list of occurrences has maximum size. For the remainder of this paper, by "motif" we refer to a 2-motif.

Given two motifs M_1 and M_2 of respective sizes $m_1 \times n_1$, and $m_2 \times n_2$, with $m_1 \leq m_2$ and $n_1 \leq n_2$, $M_1 \preceq M_2$ holds if $M_1[i,j] \preceq M_2[i+c, j+d]$ for $1 \leq i \leq m_1$ and $1 \leq j \leq n_1$, for some fixed c and d, with $0 \leq c \leq m_2 - m_1$ and $0 \leq d \leq n_2 - n_1$. We also say in this case that M_1 is a *sub-motif* of M_2, and that M_2 *implies* or *extends* or *covers* M_1. Consequently, every element in M_1 should be more general or equal to the corresponding element in M_2.
It is easy to see that the number of motifs with don't cares grows exponentially with the size of the input. This makes the exhaustive extraction of such motifs unfeasible. The following notions of maximality and irredundancy serve the purpose of reducing the size of the candidate motifs to be taken into account in the patch-dictionary generation.

Let M_1, M_2, …, M_f be the motifs in an image I. A motif M_i is *maximal in composition* if does not exist another motif M_l which occurs exactly when M_i occurs and such that M_i is smaller or more general than M_l. Consequently, there exist no M_l, $l \neq i$, with $\mathcal{L}_{M_i} = \mathcal{L}_{M_l}$ and $M_i \preceq M_l$. A motif M_i, maximal in composition, is also *maximal in length* if and only if there exists no motif M_j, $j \neq i$, such that M_i is a sub-motif of M_j and $|\mathcal{L}_{M_i}| = |\mathcal{L}_{M_j}|$. A *maximal motif* is maximal both in composition and in length.

A maximal motif M in I, with location list $\mathcal{L}_M$, is *redundant* if there exist maximal sub-motifs M_i, $1 \leq i \leq p$, such that $\mathcal{L}_M = \mathcal{L}_{M_1} \cup \mathcal{L}_{M_2} \cup \ldots \cup \mathcal{L}_{M_p}$, up to some offsets, and $M \preceq M_i$, $1 \leq i \leq p$ (i. e., every occurrence of M on I is already covered by one of the motifs M_1, M_2, …, M_p). M is also said to be *implied* by $M_1, M_2, \ldots, M_p$. A maximal motif that is not redundant is called an *irredundant* motif.

Given an image I on an alphabet Σ, let $\mathcal{M}$ be the set of all maximal motifs on I. A subset $\mathcal{B}$ of $\mathcal{M}$ is called a *basis* of $\mathcal{M}$ if the following hold:

- for each $M \in \mathcal{B}$, M is irredundant with respect to $\mathcal{B} - \{M\}$;
- let $G(\mathcal{X})$ be the set $\mathcal{M}$ of all the redundant maximal motifs implied by the set of motifs $\mathcal{X}$: then $\mathcal{M} = G(\mathcal{B})$.

More simply, let's consider the set $\mathcal{M}$ of the maximal motifs of I. We define *basis* a subset of $\mathcal{M}$ such that each motif is irredundant with respect to the others. Furthermore, from the motifs of the *basis* it is possible to generate all the other redundant motifs of I. The following theorem holds [2].

Theorem 3.1. The basis $\mathcal{B}$ of k-motifs for the image I on an alphabet Σ is unique for any k.

The following operators are also needed for our purposes. Let σ_1 and σ_2 be 2 characters on the alphabet $\Sigma \cup \{\circ\}$. The *consensus* $\sigma_1 \oplus \sigma_2$ of σ_1 and σ_2 is defined as:

$$\sigma_1 \oplus \sigma_2 = \begin{cases} \sigma_1, & \text{if } \sigma_1 = \sigma_2 \\ \circ, & \text{otherwise} \end{cases}$$

Analogously, let I_1 and I_2 be 2 images on $\Sigma \cup \{\circ\}$ s.t. $|I_i| = m_i \times n_i$ ($i \in \{1,2\}$). The *consensus* of I_1 and I_2 is obtained by overlaying the two images and replacing the *don't care* for each position, if the characters of the two images at that position are different, or the same solid character if the characters of the two images are identical at that position. The *consensus* is denoted by $C = I_1 \oplus I_2$ of size $m \times n$, where $m = min\{m_1, m_2\}$ and $n = min\{n_1, n_2\}$ and such that $C[h,k] = I_1[h,k] \oplus I_2[h,k]$, for $1 \leq h \leq m$ and $1 \leq k \leq n$.

Let I_1 and I_2 be 2 images on $\Sigma \cup \{\circ\}$, and let C be their consensus. Deleting all the external rows and columns only made up of don't cares from C yields a (possibly empty) image that is the *meet* of I_1 and I_2, denoted by $M = [I_1 \oplus I_2]$.

Given an image I on $\Sigma \cup \{\circ\}$, the set of *autocorrelations* of I is the collection of the meets generated by I with its bites. The autocorrelation generated by I and S_{ij} is denoted by A_{ij}.

To entirely cover the image, the selection of the bite is carried out starting from the bottom right of the image and then gradually moving from right to left and from the bottom upwards; the motifs of the *consensus* are calculated by superimposing these bites to the image from the top left. The same thing is repeated by taking the bites from the angle at the bottom left and then gradually moving from left to right and from bottom to top; the bites are gradually superimposed to the image starting from the upper right in order to determine the other motifs of the *consensus*. Each time that a consensus is computed, the external rows or columns made up of only *don't care* characters are removed from it in order to obtain a *meet* of the image.

The size of an autocorrelation is at most equal to the size of the bite it comes from (this happens only if the *consensus* does not have rows or columns with only *don't care* characters, otherwise the size of the meet will be less than the size of the bite).

Considering a basis of 2-motifs for an image I and the set A of autocorrelations of the same image, the basis is a subset of A and it is unique for each value of the quorum q.

Furthermore, the number of 2-motifs of a basis is linear in N. It is formally expressed by the following theorem:

Theorem 3.2. ([2]) Let $\mathcal{B}$ be a basis of 2-motifs for the image I of size N on an alphabet Σ, and $\mathcal{A}$ be the set of the autocorrelations of I, then:

- $\mathcal{B} \subseteq \mathcal{A}$
- the number of motifs in $\mathcal{B}$ is $O(N)$

3.5.2 Algorithms and implementation

The bound of Theorem 3.2 supports efficient algorithms for the extraction of the basis in $O(N^3)$ and $O(N^2)$ time, respectively, from images over general and binary alphabets [2, 59]. The top level of these algorithms is recaptured in Figure 3.5. As an example, assume as input the binary image displayed in Figure 3.6(a). Figure

```
2D BASIS EXTRACTION
Input:
- an image $I$
Output:
- the 2D basis $\mathcal{B} = \{M_1, M_2, \ldots, M_k\}$
   begin
1:    compute the set $\mathcal{A}$ of the autocorrelations of $I$
2:    eliminate from $\mathcal{A}$ possible duplicates
3:    for each $M_i \in \mathcal{A}$
4:          compute the list of occurrences $\mathcal{L}_{M_i}$
5     $\mathcal{T} = \mathcal{A}$
6:    for each $M_i \in \mathcal{T}$
7:       if there exist $p$ motifs $M_1, M_2, \ldots M_p \in \mathcal{A}$ such that $\mathcal{L}_{M_i} = \bigcup_{j=1}^{p} \mathcal{L}_{M_j}$
8:          $\mathcal{T} = \mathcal{T} - \{M_i\}$
9:    $\mathcal{B} = \mathcal{T}$
   end
```

Fig. 3.5. Illustrating the 2D BASIS EXTRACTION procedure

3.6(b) shows the consensus between I and its bite S_{13}, from which the autocorrelation A_{13} shown in Figure 3.6(c) is generated. The other autocorrelations shown in Figure 3.6(b)(c-h) are extracted similarly. We also have $A_{41} = ||0\ 1||$, and the two autocorrelations A_{12} and A_{42} consist only of the character 0, while A_{23} and A_{33} consist only of the character 1.

Once that the autocorrelations have been extracted from I, steps 2 and 3 compute their occurrence lists, and after that only irredundant motifs will be kept in the basis (steps $5 - 8$). In our example, the resulting basis is $\mathcal{B} = \{A_{31}, A_{12}, A_{21}, A_{13}\}$. Note that the basis contains only 4 of the 9 distinct autocorrelations.

The next step consists in building the dictionary of patches, ordered according to an estimate (see, e.g., [1]) of the gain induced by each patch in compression. Figure 3.7 summarizes the DICTIONARY COMPUTATION. In particular, steps 2 and 3 require $O(N)$ time, thus the overall cost in time of 1-3 is $O(N^2)$. Then, starting

$$I_{[3,3]} = \left\| \begin{matrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{matrix} \right\| \quad C_{13} = \left\| \begin{matrix} \circ \\ 1 \\ \circ \\ 1 \end{matrix} \right\| \quad A_{13} = \left\| \begin{matrix} 1 \\ \circ \\ 1 \end{matrix} \right\| \quad A_{31} = \left\| \begin{matrix} 0 & 0 & 1 \\ 1 & \circ & 1 \end{matrix} \right\| \quad A_{12} = \left\| \begin{matrix} 0 & \circ \\ 1 & 1 \\ 0 & \circ \end{matrix} \right\|$$

(a) (b) (c) (d) (e)

$$A_{21} = \left\| \begin{matrix} \circ & 1 \\ \circ & 1 \\ 0 & \circ \end{matrix} \right\| \quad A_{32} = \left\| \begin{matrix} 0 & \circ \\ \circ & 1 \end{matrix} \right\| \quad A_{22} = \left\| \begin{matrix} \circ & 1 \\ 0 & \circ \end{matrix} \right\|$$

(f) (g) (h)

Fig. 3.6. (a) An image I. (b) The consensus between I and S_{13}. (c-h) Other autocorrelations of I.

from the patch $\overline{M}$ inducing the maximum gain, the occurrence lists of the other patches are updated by discarding those occurrences that overlap with some occurrences of $\overline{M}$, and this process is repeated until all patches in the basis with a not yet empty occurrence list are processed. Steps $4 - 6$ can be performed in time $O(N^3)$. The dictionary $\mathcal{D}$ at the outset of DICTIONARY COMPUTATION is an ordered set of patches, each with those of its occurrences that will be used in the encoding.

```
DICTIONARY COMPUTATION
Input:
- an input image $I$
- the 2D basis $\mathcal{B}$ of I
Output:
- the dictionary of patches $\mathcal{D}$
    begin
1:      for each $M_i \in \mathcal{B}$
            begin
2:              compute the gain $G_i$ of $M_i$
3:              store $M_i$ in $\mathcal{D}$ in a list sorted according to $G_i$
            end
4:      for each $M_i$ in $\mathcal{D}$
5:          for each $M_j$ that follows $M_i$ in $\mathcal{D}$
6:              eliminate from $\mathcal{L}_{M_j}$ possible occurrences overlapping with some $(h.k) \in \mathcal{L}_{M_i}$
7:      for each $M_i$ in $\mathcal{D}$
8:          if $\mathcal{L}_{M_i} = \emptyset$
9:              $\mathcal{D} = \mathcal{D} - \{M_i\}$
    end
```

Fig. 3.7. The DICTIONARY COMPUTATION procedure

Figure 3.8 outlines the compression procedure, that outputs a file of intermixed cleartext and pointers ordered according to their row-major appearance in I. The overall cost of COMPRESSION ALGORITHM is dominated by the $O(N^3)$ time of DICTIONARY COMPUTATION. Figure 3.9 shows how decompression can be carried out; various kinds of interpolation may be added at the end to fill the gaps.

3.6 Experimental results

We report preliminary experimental results obtained by applying our technique to both real and randomly generated bitmap images. We implemented the BASIS EXTRACTION procedure by following the incremental paradigm described in [2], and selected the patches in our final dictionary based on the following gain estimate from [1]:

$$G_i = N - |Mi| \cdot |\mathcal{L}_{M_i}| + |Mi| + \chi \cdot (|\mathcal{L}_{M_i}| - 1),$$

```
COMPRESSION ALGORITHM
Input:
- an image $I$
Output:
- a multiset $F$ of both pixels and pointers
- a set of patches $\mathcal{D}$
   begin
1:    call BASIS EXTRACTION on $I$ and obtain $\mathcal{B} = \{M_1, M_2, \dots, M_k\}$
2:    call DICTIONARY COMPUTATION on $\mathcal{B}$ and $I$ and obtain $\mathcal{D} \subseteq \mathcal{B}$
3:    $F \leftarrow I$
4:    for each $M_i$ in $\mathcal{D}$
         begin
5:          store in $\mathcal{D}$ the pair $< P_i, i >$ of the patch $P_i$ corresponding to $M_i$ and a pointer $i$
6:          substitute in $F$ each occurrence of $M_i$ with the pointer $i$
         end
   end
```

Fig. 3.8. Illustrating the COMPRESSION ALGORITHM

```
DECOMPRESSION ALGORITHM
Input:
- a multiset $F$ of both pixels and pointers
- a set of patches $\mathcal{D}$
Output:
- an image $I$
   begin
1:    for every patch $P_i$ in $\mathcal{D}$
2:       substitute the pointer $i$ in $F$ with $P_i$
3:    $I \leftarrow F$
4:    (Optional) interpolate $I$
   end
```

Fig. 3.9. Illustrating the DECOMPRESSION ALGORITHM

where χ represents the size of a pointer, assumed to be a constant. For instance, the gain estimated for the motif A_{13} of the basis shown in Section 3.5.2, with size 3 and three occurrences, is:

$$G_i = 12 - 3 \cdot 3 + 3 + \chi \cdot (3 - 1) = 6 + 2\chi.$$

In order to control the loss, a prescribed maximum density of don't cares is adopted during the generation of the 2D basis. With general alphabets, our experiments showed an optimal setting of 3.5 for the maximum allowed density, measured as the ratio between the number of don't cares in a patch and the size of the patch. For binary images, a smaller density of 0.5 seemed to yield the best results, perhaps due to the large number of repetitions in the input image.

For comparison, we also run our basic scheme with linear patches and solid patches. The linear patches were obtained by setting to 1 the first dimension of the motifs to be extracted. The solid patches were extracted according to [3].

We ran our tests on two different kinds of data. First we generated 100 random images of integers over four different alphabets, as shown in Table 3.1. We compared our results with those returned by GZIP and BZIP. For each method, the table lists the compression ratio, the percentage of improvement (%IMPROV) over the other methods, and the loss:

$$\%\text{LOSS} = \sum_{h=1}^{N} (p_h - pe_h)/N,$$

where p_h and pe_h are the values of the colors in the h-th pixel of I and of the image I_e obtained by interpolating the image resulting from decompression.

	SOLID PATCHES	LINEAR MOTIFS	2D MOTIF BASIS	GZIP	BZIP
		512 colors			
%COMPR	46.28	36.68	31.2	36.25	36.49
%IMPROV	15.08	5.48	–	5.06	5.3
%LOSS	–	0.003	1.990	–	–
		256 colors			
%COMPR	33.76	29.68	20.58	29.67	29.53
%IMPROV	13.18	9.1	–	9.09	8.95
%LOSS	–	0.001	4.735	–	–
		128 colors			
%COMPR	27.33	22.46	13.05	24.63	22.42
%IMPROV	14.29	9.41	–	11.59	9.37
%LOSS	–	0.006	6.6	–	–
		binary			
%COMPR	7.04	6.39	5.23	6.23	5.32
%IMPROV	1.81	1.16	–	1	0.09
%LOSS	–	1.15	0.001	–	–

Table 3.1. Results on 100 random integer images generated over alphabets of different sizes

Table 3.1 shows that exploiting 2D motif basis as patches leads to a higher compression rate than with linear or solid patches (even though no limit on density was set on linear patches), and that this approach improves also over GZIP and BZIP at the expense of negligible loss.

We performed a second series of tests on bitmap images. Table 3.2 shows the results obtained on the three bitmap images: Lena, Satellite and Texture, displayed in Figure 3.10. In this case we can compare our technique also with JPEG[1]. The last three columns of the table contain the compression ratios of JPEG for different values of its quality parameter. The improvement and loss percentages for 2D motif basis are shown in Table 3.3, where the last two columns represent the percentage of loss for both linear and 2D patches. Also in this case, quantitative results show that our approach leads to a higher compression rate than with linear or solid patches and that improvements are obtained also over GZIP, BZIP and JPEG at the expense of negligible loss.

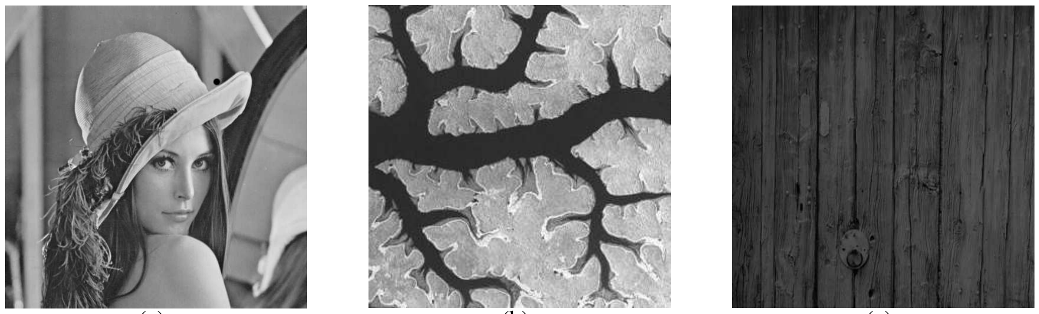

(a) (b) (c)

Fig. 3.10. (a) Lena. (b) Satellite. (c) Texture

[1] JPEG cannot be applied to arrays of integers but only to real images in the conventional formats

	SOLID PATCHES	LINEAR MOTIFS	2D MOTIF BASIS	GZIP	BZIP	JPEG		
						90	80	70
Lena	92.00	52.00	21.00	88.00	68.00	44.00	39.00	36.76
Satellite	92.68	60.97	36.58	56.09	46.34	39.02	31.70	29.26
Texture	80.36	42.39	29.69	43.74	33.54	38.25	35.8	34.6

Table 3.2. Compression percentage of the considered methods on three sample real images

	% IMPROV							% LOSS	
	SOLID PATCHES	LINEAR MOTIFS	GZIP	BZIP	JPEG			LM	2DMB
					90	80	70		
Lena	71	31	67	47	23	18	15.76	0.1	3.17
Satellite	58.1	24.39	19.51	9.76	2.44	-4.88	-7.32	0.074	1.085
Texture	50.67	12.70	14.05	3.85	8.56	6.11	4.91	0.12	3.65

Table 3.3. The percentage of improvement in the compression of 2D MOTIF BASIS w.r.t. the other techniques and its percentage loss

4

Conclusion to Part I

Part I of this thesis presented two applications of pattern extraction from images: a new image segmentation approach and a new image compression technique.

Image segmentation is graph-based and employs genetic algorithms. The fitness function extends the normalized cut criterion and the new concept of nearest neighbor for a pixel takes into account not only the spatial location of that pixel, but also the affinity with the other pixels contained in the neighborhood. The locus-based representation of individuals, together with the adopted fitness function, revealed particularly apt to deal with images modeled as graphs. In fact, as experimental results showed, this approach is able to segment images in a number of region patterns that well adhere to the human visual perception. Consequently, it could be successfully used as a preliminary step for *Object Recognition*.

On the other hand, image compression techniques based on prudently selected 2D patches are seen to convey performances that compare favorably, both in the lossless as well as in the lossy variants, with those of the corresponding more traditional linear methods. The methods here presented are inherently offline, and hence are best fit to the compression of image archives rather than in the context of transmission. By virtue of the duality between compression and pattern discovery, the patches extracted by this algorithm could be used in *Image Retrieval*.

Part II

Summary

Part II of this thesis will provide a new technique for pattern extraction from political data which evaluates the voting behavior in Italian Parliament during the XVI legislature.

In particular, the roll calls of the Italian Parliament are studied by employing multidimensional scaling, hierarchical clustering and network analysis to find voting patterns. In order to detect changes in voting behavior, the roll calls have been divided in seven periods of six months each. All the employed methods point out an increasing fragmentation of the political parties endorsing the previous government that culminated in its downfall. By using the concept of modularity at different resolution levels, the community structure of Parliament and its evolution are identified in each of the considered time periods. The performed analysis reveals as a valuable tool in detecting trend patterns and drifts of Parliamentarians. It shows its effectiveness at identifying political parties and at providing insights on the temporal evolution patterns of groups and their cohesiveness, with no need of prior knowledge about political membership of Representatives.

5

Analyzing Voting Behavior in Italian Parliament: Group Cohesion and Evolution

5.1 Introduction

In the last years political parties in Italy have been affected by a steady fragmentation, with a high number of Parliamentarians leaving the group that allowed them to be elected to join another one, often changing party many times.

In this chapter we investigate Italian Parliament by using different tools coming from Data Mining and Network Analysis with the aim of characterizing the modifications Parliament incurred in, without any knowledge about the ideology or political membership of its Representatives, but relying only on the votes cast by each Parliamentarian. We consider the roll calls of the period of three years and an half from April 2008 until October 2011, after which there was the fall of the center-right coalition that won the elections. This period has been equally divided in seven semesters and the votes cast by each Parliamentarian have been stored. Note that in our analysis we do not consider the Italian Senate, but only the House of Representatives.

The chapter is organized as follows. Section 5.2 gives a brief overview of the state-of-the-art applications of computational techniques to political field together with our personal contribution to this research field. Then, in section 5.3, an informal description of the techniques used for analyzing the voting records in Italian Parliament is presented. In section 5.4 the methodology adopted for our analysis is formally introduced. In particular, in section 5.4.1 we give a brief description of the Italian Parliament organization and the data set used for the analysis. In section 5.4.2 we describe the application of multidimensional scaling approach to the voting records. In section 5.4.3 the similarity metric used is defined, and the groups obtained by applying hierarchical clustering and community detection are discussed. Section 5.4.4 argues about the results obtained for the last semester.

5.2 State of the art in computational techniques to political field

In the following section, an overview of the most meaningful contributions to the state of the art is reported. In particular, each paragraph will describe details about each presented approach.

5.2.1 Parliamentary group cohesion and voting similarity in the Finnish Parliament

The investigation of voting records with computational techniques is not new. One of the first paper is that Pajala et al. [51], where the authors analyze the Finnish Parliament in year 2003.

The Finnish Constitution does not have political parties; the activities are so mined by only parliamentary party groups. Rules and sanctions for breaking the rules must be written from party groups. The Finnish Parliament is composed of 200 members elected for a four-year term. The vote of each Parliamentarian can be expressed as 'yes', 'no' or 'abstain'. In most cases, the approval of a legislative proposal or other issues is performed without a vote. However, usually, the important votes, such as the final considerations of law proposals, are roll-call votes. All the vote cast by each Parliamentarian are recorded and available at the Parliament's website. Analyses here include all votes expressed during the 2003 parliamentary year. Furthermore, various voting thresholds are provided: the first one is a simple majority of the votes cast. This means that a proposal can be approved only when the number of ’yes’ votes is greater than the number of ’no’ votes. The second one is for a few special cases and consists in a 2/3 majority from the votes cast which is required for the approval. The last modality can be adopted when constitutional change is to be declared urgent and requires a 5/6 majority for the approval. Let’s recall that 2003 was characterized only by simple majority votes. Now, it’s particularly important to summarize the political situation in 2003 inside the Finnish Parliament.

In March 2003 elections changed the situation of the cabinet with the decline of the ’rainbow’ coalition (KOK, SDP, VAS, VIHR, R). From the previous cabinet parties, SDP obtained two seats (into 53 seats), KOK lost six seats (40), while the main opposition party KESK won the elections obtaining seven more seats (55). Consequently, KESK proposed the new prime minister, and SDP and R became the other two cabinet parties. However, R lost three seats (9). Thus, a majority took place in the Finnish Parliament. Considering this political situation, authors perform some kinds of analysis.

First of all, they compute the *agreement index* proposed by Hix et al. [32] for assessing the cohesion inside the parliamentary party groups. They show as the most cohesive group is the Swedish people’s party R (.965), together with the other two cabinet groups KESK (.961) and SD (.921). The opposition groups are less cohesive, and only PS (.948) is comparable with the cabinet groups. Other groups KD (.866), VAS (.861), VIHR (.851) and KOK (.817) exhibit lower cohesion values.

Secondly, they consider the roll calls and the votes cast by each of the Parliamentarians to compute a dissimilarity matrix between every pairs of Parliamentarians, based on Rajski’s distance that uses *mutual information* and *joint entropy*. The lower the Rajski’s distance between two Parliamentarians, the greater their similarity. They used the agglomerative hierarchical clustering algorithm *agnes* [38] with the average linkage method, and built some dendrograms. All the Parliamentarians are subdivided into clusters by the hierarchical clustering, so that each Parliamentarian corresponds to a given cluster. The next step consists in finding from the algorithm the closest pair of clusters and merges them into a single one. The average linkage method sets the distance between two clusters A and B as the average distance among all pairs of Parliamentarians from A and B, so that one Parliamentarian in the pair is a member of A, and the other of B. Hierarchical clustering employed on their distance matrix allowed to distinguish quite precisely between the cabinet coalition and the opposition. They found that the analysis performed is able to capture the main characteristics of the Finnish Parliament.

5.2.2 Community structure in the United States House of Representative

Another interesting study regarding the United States House of Representatives from 101st-108th Congresses has been done by Porter et al. [58]. They defined bipartite collaboration networks by the assignments of Representatives to House committees and subcommittees. Each edge in the network between two (sub)committees has a weight which corresponds to the normalized interlock. The interlock between two committees is equal to the number of their common members. The normalization is obtained considering the committee sizes by dividing the interlock by the expected number of common members if assignments were defined independently and uniformly at random. Thus, links between pairs of subcommittees or pairs of committees are the normalized degree of joint membership between (sub)committees and links between committees and subcommittees are considered as the fraction of standing committee members on subcommittees. However, it is quite difficult to visualize and analyze the two kinds of networks of committee assignments. Consequently, the authors adopted a different kind of network visualization consisting in a projection of the network onto either the committees or the Representatives: a network is built whose nodes are the committees and whose edges represent common membership or interlocks between committees.

Then, they investigated the hierarchical and modular structure of these networks by using different community detection methods. So, they adopted various methods of hierarchical clustering, in which, from the network, a hierarchical tree structure is defined. One of these methods is known as single linkage clustering. To implement single linkage clustering, the authors start with the complete set of committees for a given Congress. Then, they merge committees sequentially starting with the pair with the highest normalized interlock value, followed by the next highest, and so on. This process is able to construct clusters of committees, which can be represented by using a tree or dendrogram. From this analysis, four hierarchical levels of clustering have been extracted: subcommittees, standing and select committees, groups of standing and select committees, and the entire House. These single linkage clustering dendrograms revealed also an organization corresponding to groups of subcommittees inside larger standing committees. In order to perform an analysis of the obtained hierarchies in the House committee networks, authors used the modularity concept, modified to mine committee weighted networks. In fact, the projected one-mode networks are weighted. Instead of counting numbers of edges falling between particular groups, they count the sums of the weights of those edges. This concept of modularity measures when a particular division of the network has more edge weight within groups than one would expect on the basis of chance and it is used to evaluate the efficacy of the organizational grouping of the networks and to compare the dendrograms to each other. Furthermore, the community structure of the network of committees has been explored by using three other methods: two based on betweenness values computed on the full bipartite networks of Representatives and committees and a local community detection algorithm for weighted networks. In this way, the authors identified connections between committees and correlations among committee assignments and Representatives' political positions. Furthermore, changes in the network structure corresponded to change of Senate majority from Democrats to Republicans.

Moreover, they applied SVD to evaluate the House roll call votes. Here authors apply the multidimensional scaling technique known as singular value decomposition (SVD) in order to analyze voting records of each session of the House. An $n \times m$ voting matrix B is defined where each row represents each of the n Representatives in the House and each column represents each of the m votes taken during a two-year term. A generic element B_{ij} can be $+1$ if the Representative i voted "yes" on measure j and -1 if (s)he voted "no". If a Representative did not vote because of absence or abstention, the corresponding element is 0. The SVD analysis is able to identify groups of Representatives who voted in a similar fashion on some measures. An approximation of the Representatives voting records can be characterized by just two coordinates. The first one is called "partisan coordinate" and is able to measure the polarization degree for members of the two major parties. The second one is called "bipartisan coordinate" and expresses how often a Representative votes with the majority. From this analysis, it is possible to observe as Democrats are grouped together and are almost completely separated from Republicans.

5.2.3 Community structure in congressional cosponsorship networks

Zhang et al. [80] studied the United States Congress by building networks for Members of Congress. In these "bipartite" networks, a Member of Congress is linked by an edge to each sponsored or cosponsored bill. This kind of network is represented by using a bipartite adjacency matrix M, whose generic element M_{ij} is equal to 1 if legislator i (co-)sponsored bill j and 0 otherwise. This means that there are two types of nodes which are Congressmen and bills, while the edges in the network represent sponsorships or cosponsorships. By using information about these Congressional committee and subcommittee assignments, authors create another kind of network. Inside the bipartite adjacency matrices built from these networks, a generic element M_{ij} indicates the assignment of Representative i to committee or subcommittee j. Analysis is focused on cosponsorship networks obtained by using "unipartite" projections. In these networks, the nodes are legislators and the weighted edges linking them indicate how many bills they together (co-)sponsored.

Network communities are detected by using the concept of modularity. Following this concept, a community should have more internal connections among its nodes than connections between its nodes and those in other communities. Modularity is evaluated for chosen partitions of the network and it measures the degree of the interactions which are detected between legislators within the identified communities rather than across them.

Each network is so recursively partitioned in order to generate trees or dendrograms to assess its hierarchical structure. This process is able to discover communities of various sizes by iteratively clustering the legislators by using the partitioning algorithm. Modularity evaluates the number of intra-community versus inter-community links for a given partition, consequently it has been adopted to quantify the growth in polarization in the U.S. Congress. In particular, during the considered period of 24 years, from the 96th to 108th Congresses, an increase in modularity has been obtained. This corresponds to an increase in party polarization of the Congress that caused the control by the Republicans of both chambers.

Authors used also a multidimensional scaling technique called NOMINATE and singular value decomposition analysis (SVD). They show that a matrix of roll call votes can be approximated by using two coordinates: a generic liberal-conservative dimension and a second social dimension. However, the same approaches demonstrate that multiple dimensions are needed to adequately approximate a matrix of cosponsorships. The adopted eigenvector methods detect large communities corresponding to known political cliques. It can be shown that Members of Congress with similar ideologies are clustered together in the identified communities.

5.2.4 Party polarization in Congress: a network science approach

In [73] the polarization in the United States Congress is evaluated by using also the concept of network modularity. Each node represents a legislator in the network and each edge is the level of agreement between two legislators in roll-call voting, indicating the average number of equal votes between them. Generally in a legislature, groups like parties contain strong connections between legislators within the same group but relatively weak connections between individuals in different groups.

Authors created similarity matrices from the network, describing voting similarities among legislators in a single Congress of the House of Representatives or Senate. Each of these networks is thus represented by an $n \times n$ matrix A, where n is the number of legislators and a generic element A_{ij} represents the fraction of equal votes for the two legislators.

Multiple community-detection algorithms are employed on the similarity matrices of legislators to identify groups that maximize the modularity inside each roll-call network for both the Senate and the House of Representative. In particular, given a partition of the network into communities, the modularity Q measures the fraction of total connections embedded within the specified communities minus the expected total strength of such connections. Modularity is able to assess the quality of community partitioning, consequently, partitions with higher modularity are more polarized. The community partition that maximizes modularity for each Congress is called ”maximum modularity partition”.

Modularity is adopted for measuring the degree of polarization, revealing the main political groups and the divisions among them. A non-monotonic relationship between maximum modularity and a consequent majority party switch are explored, demonstrating that the changes in majority are more likely when the modularity value is moderate, uncommon otherwise. In particular, modularity values in Congress t are used to predict modifications in the majority party for Congress $t + 1$. A non-monotonic relationship between modularity and the stability of the majority party is found in both chambers of Congress. When modularity is low, a change in majority control seems to be less likely; at high levels of modularity, the minority cannot overcome the majoritys cohesion. In both of these cases, it is infrequent to have majority-party switches. However, when modularity exhibits medium values, this corresponds to changes taking place for majority cohesion and to a less stable party system.This is called ”partial polarization” hypothesis.

At the individual-level, some measures associated with modularity, called “divisiveness” and “solidarity” are computed to predict the reelection success for individual House members. The divisiveness measures the effect that each legislator could have on the aggregate polarization of his legislature by using roll-call adjacency matrices. About solidity, when its value is close to 1, the legislator and community are strongly aligned; otherwise, when the solidarity is close to 0, the legislator is not strongly aligned with his or her community. Performing this kind of analysis, authors find that divisiveness has a negative influence on reelection chances and that group solidarity has a positive influence. Furthermore, divisiveness is associated with decreased reelection probability, and the combination of divisiveness and solidarity has a significant positive impact on reelection.

5.2.5 Community structure in the United Nations General Assembly

Macon et al. [47] investigated the community structure of networks constructed from voting records of the United Nations General Assembly. The UNGA was founded in 1946. Annual sessions from 1946 to 2008 are considered and unanimous votes are removed from the data, because they don’t give information about the network structure of voting agreements and disagreements between countries.

Three different networks are so defined. The first one is a weighted unsigned network of voting similarities, whose nodes are the countries and whose edges between pairs of countries are weighted by using an agreement measure. This represents the number of agreements on resolutions (yes-yes, no-no, or abstain-abstain) between the two involved countries. A weighted and unipartite adjacency matrix is built from such a network, by setting all the elements along the diagonal equal to zero (corresponding to the self-edges inside the voting similarity matrix). The second kind of network is constructed by considering also the number of yes-no disagreements in the elements of the voting similarity matrix. The last kind of network is a signed bipartite network of countries voting for individual resolutions. A bipartite network is composed of two types of node, countries and resolutions, and each edge is a link between a node of one type and a node of the other. For each session, the adjacency matrix is defined from the bipartite network, whose elements are $+1$ if country i voted "yes" on resolution j, -1 if country i voted "no" on resolution j, 0 in case of absence or abstention.

Authors find communities on these networks by optimizing the modularity quality function and some of its generalizations. In particular, they perform a comparison among the community detection results obtained by partitioning the different network representations in communities on each of the considered sessions of the UNGA. Partitions of the network of agreements optimizing the modularity are considered at two different values of the modularity resolution parameter ($\gamma_1 = 1$ and $\gamma_2 > 1$). For each kind of network, dominant voting pattern including two large communities corresponding to majority and minority coalitions are found, for each of the considered UNGA sessions. This kind of polarization is clearer for partitions extracted by optimizing modularity of the network of agreements at the standard resolution parameter value $\gamma_1 = 1$ and of the bipartite networks. Furthermore, small and medium-size cores of coalitions in the network of agreements persist for resolution parameter value $\gamma_2 > 1$.

By analyzing the resolutions with respect to the voting agreement, the authors are able to detect historical trends and changes in the United General Assembly community structure. In fact, observations appear to be consistent with the expected East-West split of the Cold War and the North-South division of recent sessions that has been detected by social scientists using qualitative methods.

5.2.6 Analyzing the US Senate in 2003

Jakulin and Buntine in [36] presented an approach to analyze the roll calls in the US Senate in year 2003. For each roll call, the database of the legislative information contains a list of votes cast by each of the 100 senators and 459 roll calls are considered in the year 2003. For each of those, the vote of every senator is expressed by using three possible ways: "Yea", "Nay" and "Not Voting".

The first kind of analysis employes a *mutual information* based measure, which is the Rajski's distance, to compute the dissimilarity between two senators, given their votes. The greater this distance between their votes, the lower the similarity between the two senators. Given the distance measure, a graphical matrix is computed by considering the distances between all pairs of senators. Dissimilarity matrices are then sorted by using a hierarchical clustering approach, such that similar senators begin adjacent to one another. Furthermore, a graph is created by selecting only a certain number of the strongest similarities, with nodes corresponding to senators and edges to their connections. An artificial threshold is adopted to discriminate between a connection and the absence of it. The nodes are labelled with the total number of votes cast, while the edges are marked with the percentage of roll calls in which both senators cast the same vote.

Another kind of analysis evaluates the influence of a voter by postulating a Shannon information channel between the outcome and a voter.

Thirdly, principal component analysis is adopted on the roll call data. In particular, a roll call matrix M is constructed whose rows are senators, and whose columns are roll calls. If a generic element in position (j, v) is equal to 1, the j-th senator voted "Yea" in the v-th roll call, and if it is equal to -1, the vote was "Nay". If the senator did not vote, some value needs to be used nevertheless. The two new coordinates of the senators are those columns of the U matrix, (characterizing the decomposition of M), that correspond to the two highest singular values, multiplied by them. These two columns can be understood as uncorrelated latent votes that identify the ideological position of the senator. Another adopted approach for analyzing multidimensional data, such as a senators voting patterns, is the probabilistic version of principal components analysis. It substitutes the continuous valued variables with fully discrete ones. Here the full set of votes for each senator is modeled by using several voting patterns, corresponding to the components. A voting pattern gives the propensity to vote in a particular way and assumes independence between individual senators votes.

Finally, several kinds of analysis are performed based on the identification of discrete blocks. The first type of analysis investigates the cohesion within a block (some blocks may be more cohesive in the sense that the voting is more bloc-aligned), evaluated by using an *agreement index* and the dissimilarities between blocs (individual blocs can be similar or dissimilar, like senators). It is very difficult for a single senator to influence the situation, because rarely is one able to change the outcome of a roll call by one vote. However, once the the blocs voting in a similar way are detected across a number of roll calls, the influence of changed behavior of a group is analyzed. In particular, two kinds of altered behavior are considered: bloc abstention and bloc elimination. By using this approach, it is possible to obtain a list of roll calls for which it is deemed that the behavior of a bloc has affected the outcome.

5.2.7 Our contribution

Let's recall that a steady fragmentation of the political parties in Italy has taken place in the last years. In particular, during the XVI legislature, we observed a high number of Parliamentarians leaving their own group to join another one and many internal changes incurring inside the italian political parties. In particular, remember that the fragmentation of the majority center-right coalition in those years caused in Italy the government breakdown.

To the best of our knowledge, automatic tools for analyzing the political scenario inside the Italian Parliament have never been adopted beforehand. Consequently, our approach represents the first tentative of exploring the Italian Parliament by using a Data Mining and Network Analysis framework. In particular, our main contribution consists in identifying the italian political parties and the organizational structure of the Parliament and in investigating the changes inside the Italian Parliament, relying only on the votes cast by each Parliamentarian and without any a priori knowledge about the italian political situation. Furthermore, we try to keep track of the temporal evolution of the political groups and to analyze the trend of their inner cohesiveness along the time. Finally, we adopt our framework also for discovering hidden political information from the voting records.

5.3 Analysis of voting behavior in Italian Parliament

Next, we informally introduce the main aspects underlying the analysis of voting behavior in Italian Parliament and we try to give an overview of our technique.

First of all, we extract the vote of each Parliamentarian for every roll call from the Database of the Italian Parliament. After that, we store the vote cast by each Parliamentarian for each semester in order to analyze the voting records.

Voting records have been used in two different ways. In the first approach, we build from them some voting matrices, representing the matrices of the vote cast for each semester. A generic element inside the matrix can be $+1$, -1 or 0, depending on whether the Parliamentarian of row i voted 'yes' or 'no' on the measure of column j or whether the Parliamentarian did not vote on that measure (Fig. 5.1).

We start from them and directly use them to show party cohesion during the considered period, by using a party cohesion index. Then, we apply a multidimensional scaling technique to reveal political affinity of Parliamentarians, independently of their true party membership. This kind of analysis is interesting because it is able to reproduce the effective political alliances, without assuming parties as a-priori relevant clusters.

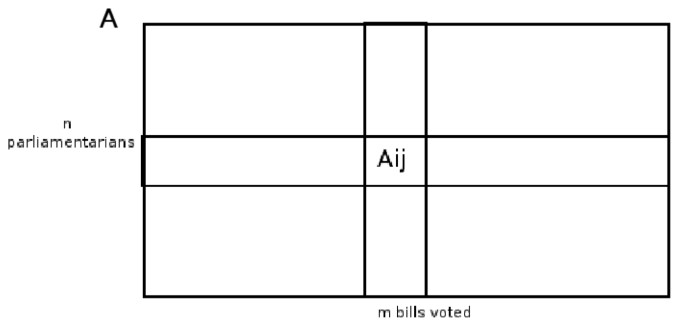


Fig. 5.1. An example of voting matrix.

In the second one, from voting records we computed similarity between each pairs of Representatives and we built some similarity matrices, one for each semester. We use them to detect structural organization and evolution of Parliament by applying hierarchical clustering. Similarity for a pair of Parliamentarians is computed by using a matching coefficient measuring the fraction of equal votes respect to the total votes they cast (Fig. 5.2).

$$SMC(p_1, p_2) = \frac{yy + nn}{yy + nn + yn + ny} = \frac{3+1}{3+1+2+1}$$

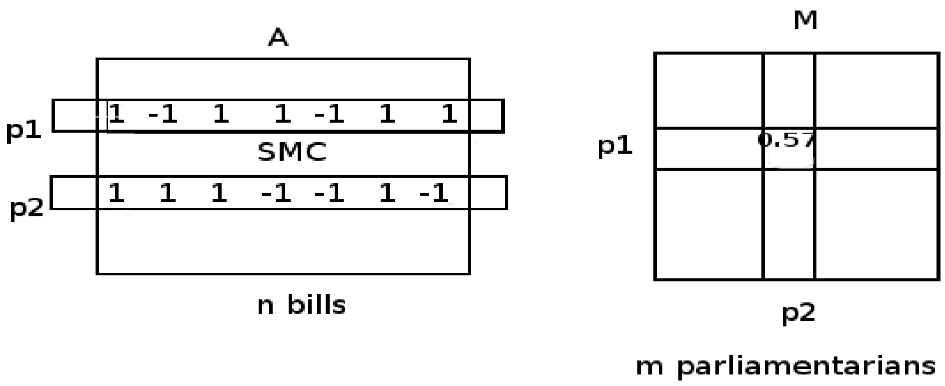


Fig. 5.2. An example of similarity matrix construction. yy represents the number of votes when both the Parliamentarians voted 'yes' on a given bill, nn when both Parliamentarians voted 'no', yn when the first one voted 'yes' and the second one voted 'no' and ny when the first one voted 'no' and the second one voted 'yes'.

Also, performing a "binarization" of the similarity matrices by using a similarity threshold value, we are able to graphically represent the similarity matrices, showing intra-group similarity, and to apply community detection techniques based on the concept of modularity. In particular, from the binary similarity matrix, we construct an undirected and unweighted network whose nodes represent the Parliamentarians. An edge between two Parliamentarian exists only if the corresponding entry inside the binary matrix is equal to 1. By using a genetic algorithm optimizing the modularity function, we are able to discover communities of Parliamentarians inside the network.

All the approaches conducted to coherent results. Hierarchical clustering is able to represent the political alliances along all the semesters. However, by using the modularity concept, we identify communities of members that voted similarly, and investigate how the cohesion of parties evolves along the semesters.

Finally, we decide to analyze the last semester separately from the other ones because of the alteration of the voting behavior of the Parliamentarians in that period. In particular, we observe as the political party organization disappears, the number of voted measures is less than the other semesters and each Parliamentarian seems to vote independently of his group. This kind of situation is shown clearly from our analysis, revealing, first of all, the loss of polarization in multidimensional scaling. This means that is no longer possible to distinguish between the two main political coalitions. Secondly, hierarchical clustering analysis discovers a unique cluster including all the parties and the visualization of the similarity matrix exhibits a high fragmentation. Furthermore, Parliamentarians coming from different parties appear as clustered together in the community detection analysis.

In conclusion, our analysis provides an explicit and clear view of the steady fragmentation of the coalition endorsing the center-right government, that caused the majority breakdown. Thus modularity allows a more deep analysis of the internal agreement of parties and demonstrated a powerful means to give insights of changes in majority party.

5.4 The methodology

In the following sections, we formally present all the concepts defining our analysis, including the used data format, the party cohesion evaluation, the multidimensional scaling and hierarchical clustering definitions and the adopted community detection framework.

5.4.1 Data description

The current Italian Parliament has been elected in April 2008 and it is constituted by 630 representatives originally elected in 5 main political parties: Popolo della Libertá (PDL), Lega Nord (LN), Partito Democratico (PD), Italia dei Valori (IDV), and Unione di Centro (UDC). The majority of center-right that governed Italy until November 2011 was composed by the first two parties. To better understand the analysis we performed, it is important to know that two main events characterized the political organization of Parliament: (1) in July 2010 a group of Representatives divided from PDL to form a new political party named Futuro e Libertá (FL); (2) in December 2010 Parliamentarians, mainly coming from the center-left coalition, separated from their party to constitute a new coalition that endorsed the center-right government, allowing it to rule the country for other almost ten months. Furthermore, along all the three years and an half, several Representatives abandoned their party to move in a group called Misto.

The Italian Parliament maintains a database of the legislative activity by storing, for each bill voted, the list of votes cast by each Representative. From the web site http://parlamento.openpolis.it it is possible to download the voting record of each Parliamentarian, together with some personal information, such as territorial origin, and actual group membership. For every roll call, the Openpolis database stores the vote of each Parliamentarian in three ways: ’yes’, ’no’, and ’not voting’. This last kind of vote can be due to either absence or abstention, but they are treated in the same manner.

Table 5.1. Number of voted measures for each semester.

I	II	III	IV	V	VI	VII
386	422	328	343	373	332	89

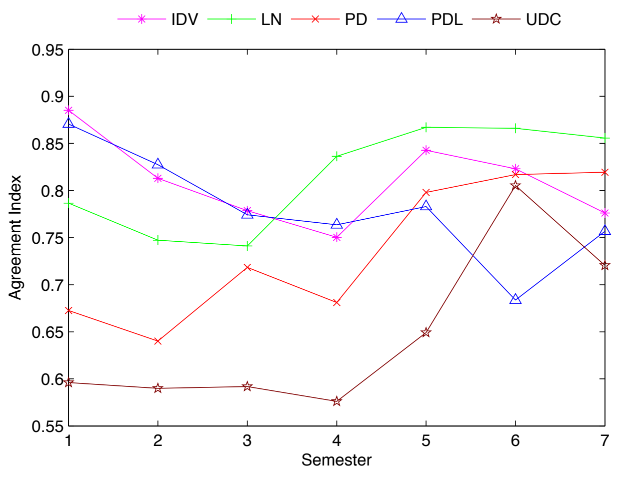
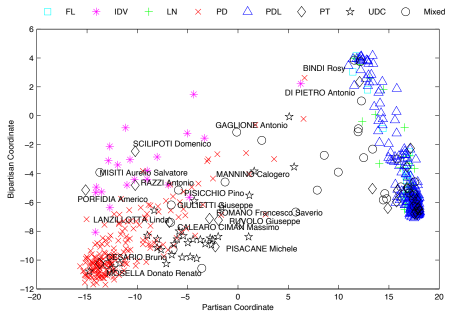
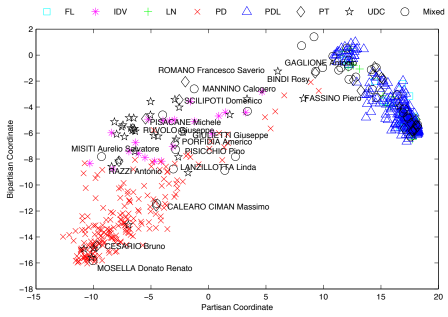


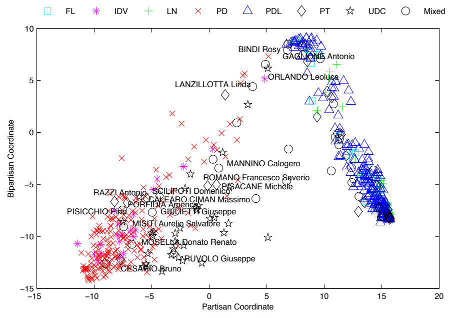
Fig. 5.3. Agreement index of parties for all the semesters.



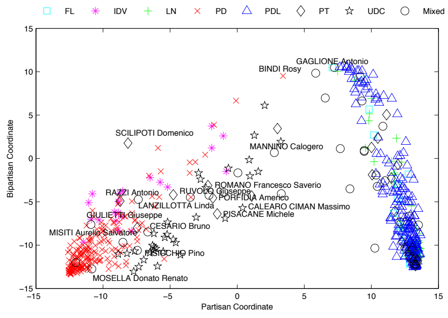
(a) I Semester:April-September 2008



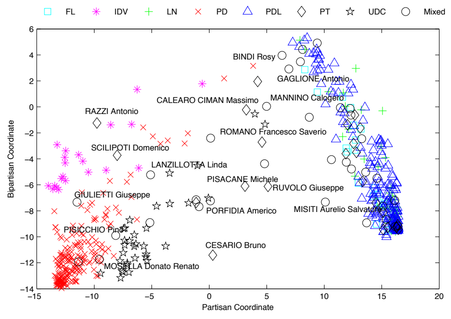
(b) II Semester:October 2008-March 2009



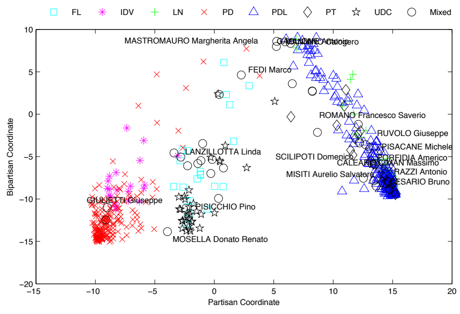
(c) III Semester:April-September 2009



(d) IV Semester:October 2009-March 2010



(e) V Semester:April-September 2010



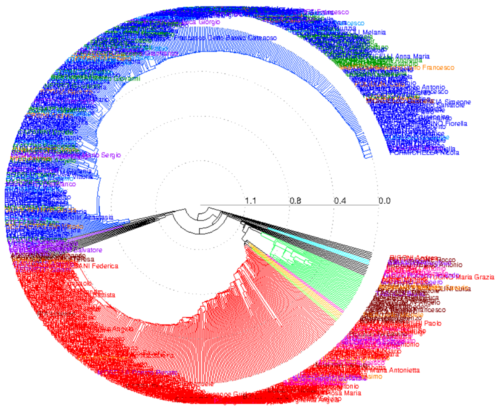
(f) VI Semester:October 2010-March 2011

Fig. 5.4. Singular value decomposition of the Italian Parliament voting behavior for each of the six semesters starting from April 2008 until March 2011.

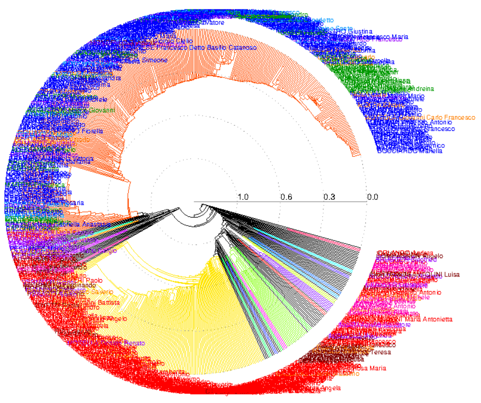
5.4.2 Analysis of voting patterns

We collected the roll calls of the Italian Parliament in the period starting from April 2008 until October 2011, after which there was the fall of the center-right coalition that won the elections. This period of three years and an half has been equally divided in seven semesters and the votes cast by each Parliamentarian have been stored in matrices of size $n \times m$, where n is the number of Parliamentarians, and m is the number of bills voted in the reference period. Since some Parliamentarians, for several reasons, never voted, they have been eliminated. Thus the number n of Representatives reduced to 612. As regards m, it assumes a different value, depending on the semester. The number of bills voted is reported in Table 5.1.

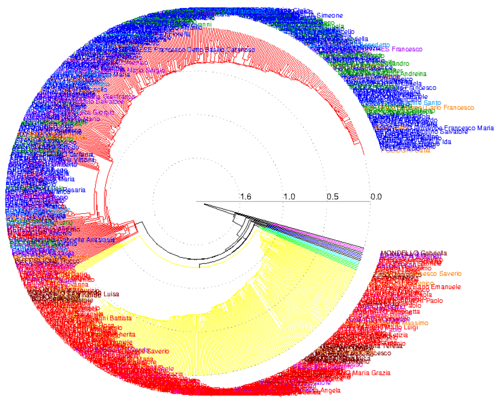
Seven voting matrices have been built in the following way: an element A_{ij} of a voting matrix A is +1 if the Representative i voted yes on measure j, -1 if (s)he voted no, and 0 if (s)he did not vote. The voting matrices are exploited to study the voting behavior of the Italian Parliament in two different ways. In the first



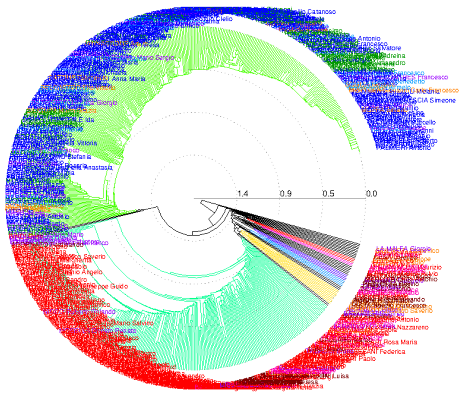
(a) I Semester



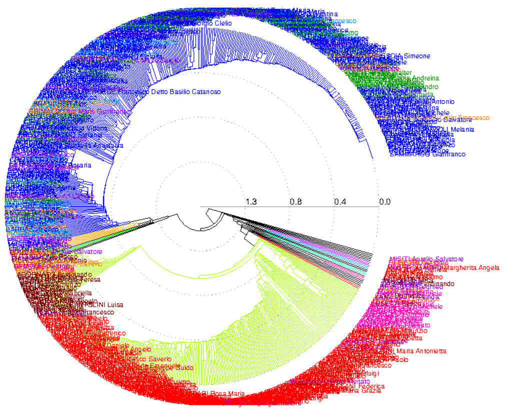
(b) II Semester



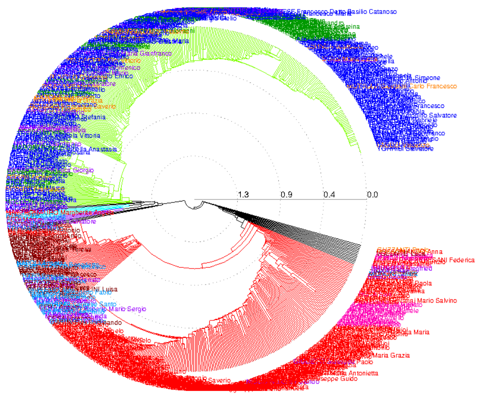
(c) III Semester



(d) IV Semester



(e) V Semester



(f) VI Semester

Fig. 5.5. Dendrograms obtained by the single linkage clustering algorithm for each semester. Internal colors correspond to the clusters found by the algorithm, external colors to the true parties. The association color-party is the following: FL: cyan, IDV: magenta, LN: green, PD: red, PDL: blue, PT: orange, UDC: brown, Misto: violet.

approach we use them to compute party cohesion and to characterize the political affinity of Parliamentarians, independently of their true party membership. In the second one, we compute similarity between each pairs of Representatives and try to detect structural organization and evolution by applying hierarchical clustering and community detection based on the concept of modularity.

Party Cohesion

Given the voting matrices, the first investigation that can be performed is to compute the cohesion of each political party along the considered period and compare the results obtained. To this end, the *agreement index* [32] measures the level of cohesion within a party by exploiting the number of equal votes for each roll call. The agreement index for each roll call is defined as follows:

$$AI_i = \frac{max\{y_i, n_i, a_i\} - \frac{y_i+n_i+a_i-max\{y_i,n_i,a_i\}}{2}}{y_i + n_i + a_i}$$

where y_i is the number of members who voted Yes in the voting i, n_i is the number of members who voted No, and a_i is the number of members who did not vote. Group cohesion is then computed as the average of agreement indices for all the roll calls: $AI = \frac{\sum_i^m AI_i}{m}$. The agreement index ranges from 0 (complete disagreement) to 1 (complete agreement). Figure 5.3 displays the trend of agreement index of the 5 main political parties during the seven semesters. It is clear from the figure that the opposition parties show an increasing cohesion, while PDL, that started with a value near to 0.9, has a constant downtrend until the sixth semester, with a slight increment in the last semester. The variation of internal cohesion well reflects the actual political situation along the considered periods.

Singular Value Decomposition

We now analyze the voting behavior of Italian Parliament by applying the well known multidimensional scaling technique known as *Singular Value Decomposition* (SVD)[67], whose advantages with respect to other techniques have been discussed in [8]. Let A be an $n \times m$ voting matrix where rows correspond to Representatives and columns to the votes cast to approve a law. The *Singular Value Decomposition* of A is any factorization of the form

$$A = U \times \Lambda \times V^T$$

where U is an $n \times n$ orthogonal matrix, V is an $m \times m$ orthogonal matrix and Λ is an $n \times m$ diagonal matrix with $\lambda_{ij} = 0$ if $i \neq j$. The diagonal elements λ_i are called the $eigenvalues$ of A. It has been shown that there exist matrices U and V such that the diagonal elements of Λ are the square roots of the nonzero eigenvalues of both AA^T and A^TA, and they can be sorted such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ [67]. Geometrically this factorization defines a rotation of the axis of the vector space defined by A where V gives the directions, Λ the strengths of the dimensions, and $U \times \Lambda$ the position of the points along the new axis. Intuitively, the U matrix can be viewed as a similarity matrix among the rows of A, i.e. the Representatives, the V matrix as a similarity matrix among the columns of A, i.e. the votes cast for each law, the Λ matrix gives a measure of how much the data distribution is kept in the new space [35]. If the singular values λ_i present a fast decay, then $U \times \Lambda$ provides a good approximation of the original voting matrix A. In particular, by projecting on the first two coordinates, we obtain a compressed representation of the voting matrix that approximates it at the best. The visualization of the projected approximation matrix, allows to identify groups of Representatives that voted in a similar way on many bills. As observed in [58], the first coordinate correlates to party membership, thus it is called the *partisan* coordinate. The second coordinate correlates to how often a Representative voted with the majority, thus it is called the *bipartisan* coordinate.

Figure 5.4 shows the application of SVD on the voting records of the Italian Parliament for the first six semesters of the current legislature. Each point corresponds to the projection of votes cast by a single Parliamentarian onto the leading two eigenvectors *partisan* and *bipartisan*. Each party has been assigned a different color and symbol. The main objective of this analysis was to study the changes in voting behavior of those Parliamentarians that moved from the opposition coalition to the majority one. Thus we selected some members of PT and Misto group, and visualized their names on all the figures. First of all we point out that the representation of the two coalitions center-right and center-left, and their evolution along the three years, summarized by the six figures, is rather impressive.

Figure 5.4(a) clearly shows a compact center-right aggregation, a less cohesive, but clearly distinguishable, center-left alliance, and a strong connected PD sub-group (left bottom). It is worth to note that this subgroup maintains its connectedness for all the time periods, with a slight dispersion in the second semester. The same cohesiveness is shown by PDL and LN, as expected. Moreover FL, which was included in PDL until July 2010, demonstrated its political disagreement in the sixth semester by coming nearer to UDC, as actually happened.

As regards the chosen members of PT and Misto groups, we can observe a steady movement from the center-left coalition to the center-right one since the fourth semester. This shift is much more evident in the 5th semester, when the voting behavior of these Representatives approached closer and closer to center-right majority. In fact, all the Parliamentarians located in the central part of Figure 5.4(e), appear at right in Figure 5.4(f), indistinguishable from the majority coalition. We also notice that there is a PD Parliamentarian positioned upper, near the right coalition, for five semesters. Because of the interpretation of the bipartisan coordinate, her location means that she mostly voted with the majority. This dissimilarity from the own political party, perhaps can be explained by the fact that this Representative is vice-president of the chamber.

Analysis of voting behavior with Singular Value Decomposition is thus a powerful tool to characterize political point of view of Parliamentarians, and to trace the evolution of their position along consecutive time periods. SVD is able to find structural patterns and latent information in the voting records without any knowledge about the political orientation of Representatives.

5.4.3 Parliamentarians similarity

There can be different ways of defining similarity between two Parliamentarians from the voting matrix. For example, Jakulin and Buntine [36] used the mutual information concept. However, as observed by the authors, if two members always vote in the opposite way, they also are considered similar. We think that this kind of proximity measure misrepresents the Representative closeness, thus we employed a different measure. Considering that when two Representatives cast a vote the values yes and no should be considered equally important in comparing their political affinity, we adopted the proximity measure known as *simple matching coefficient* (SMC) [68]. We ignored the cases when at least one of the two did not vote because, as already pointed out, this means either abstention or absence, and we cannot distinguish between them. Thus there can be four different outcomes: (1) yy, both voted yes, (2) nn, both voted no, (3) yn, the first Parliamentarian voted yes and the second one no, (4) ny, the first Parliamentarian voted no and the second one yes. Then the SMC of Parliamentarians p_1 and p_2 is defined as

$$SMC(p_1, p_2) = \frac{yy + nn}{yy + nn + yn + ny}$$

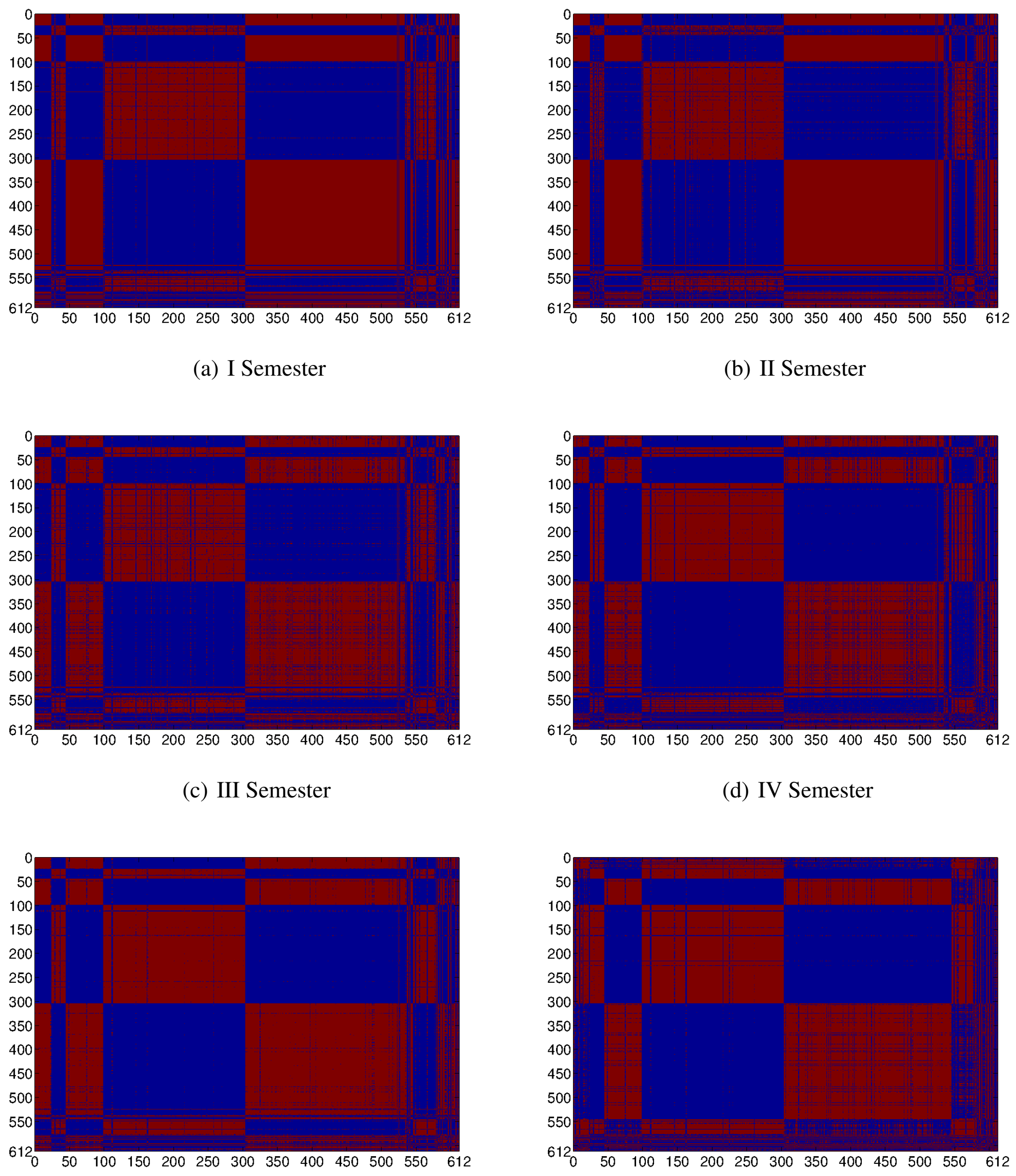
The simple matching coefficient thus computes the fraction of equal votes, both yes or no, with respect to the total votes they cast.

The defined similarity metric allows us to measure the closeness of each pairs of Parliamentarians on the basis of their voting behavior. In such a way a symmetric similarity matrix M among all the Parliamentarians can be built, and their proximity with the members of the same or opposite parties studied. A summarized view of the affinity between each couple of Representatives can be obtained in different ways. In the following we first apply a hierarchical clustering algorithm, and then we give a graphical representation of the similarity matrix.

Hierarchical clustering

We apply the agglomerative hierarchical clustering method known as *single linkage clustering* [68]. The algorithm uses the smallest distance between two Parliamentarians and it generates a hierarchical cluster tree known as *dendrogram*. The dendrogram shows the cluster-subcluster relationships and the hierarchical structure of the merged groups. Figure 5.5 represents very well the political alliances along all the semesters. The colors inside the dendrogram represent the clusters found by the algorithm. Attached to the leaves there are the names of the corresponding politicians, painted with the colors of the true associated party. For a clearer visibility, all the depicted polar dendrograms are available at the website https://sites.google.com/site/alessiaamelio/software-tools.

In Figure 5.5(a) we can observe as the two main political parties, PD in red and PDL in blue, correspond to the two main clusters of the dendrogram for all the semesters. The other parties (IDV in magenta, FL in cyan, LN in green, PT in orange, UDC in brown, and Misto in violet) are clusters of smaller size, or they are merged inside the main clusters. For example, LN party is grouped together with PDL in all the semesters,



(a) I Semester (b) II Semester

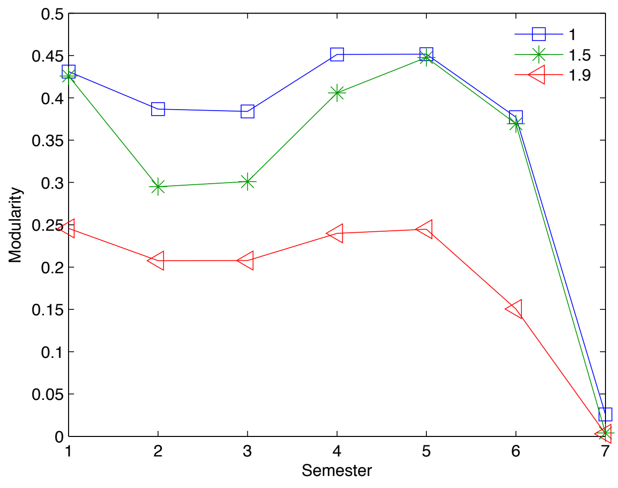
(c) III Semester (d) IV Semester

(e) V Semester (f) VI Semester

Fig. 5.6. Visualization of the binary similarity matrices sorted by party membership, for each of the six semesters. The row interval corresponding to each party are the following: FL [1:23]; IDV [24:44]; LN [45:98]; PD [99:303]; PDL [304:521]; PT [522:545]; UDC [546:578]; Misto [579:612].

reflecting the real political (center-right) alliance between PDL and LN. Another similar case is IDV: most of the members are grouped with the PD while some of them appear in different clusters for all the semesters.
Let us now consider the remaining parties. FL, as already described, was included into PDL until July 2010, when internal problems caused the movement of FL in the direction of center-left alliance. This phenomenon is captured by the clustering process. In fact FL is included into the majority for the semesters I-V (Figures 5.5(a-e)), while in the 6th semester all the members of FL are separated from PDL and grouped together with the opposite part (Figure 5.5(f)).
In order to analyze more clearly the trend of PT and Misto parties, we looked not only at the dendrograms but also at the confusion matrices generated for all the semesters. They show what really happened along the

semesters of the legislature: the gradual movement of PT and of some members of the Misto group in the direction of the center-right alliance.
Furthermore, it is interesting to observe that UDC is recognized from the clustering process as a group (Figure 5.5(a)), while in the 6th semester (Figure 5.5 (f)) it appears together with FL and grouped with PD. This is due to the political alliance between the UDC and FL and to the movement of both parties in the direction of the center-left alliance.
It is worth to note as the main voting patterns revealed by hierarchical clustering totally agree with the results of the SVD analysis illustrated in the previous section.



(a)

Fig. 5.7. Modulatity values.

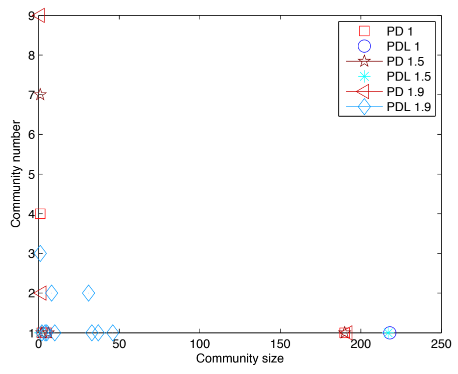
Similarity matrix visualization

In order to visualize the similarity matrix M, a binarized matrix B has been built from M by assigning 1 to the element B_{ij} if $M(i, j) \geq 0.6$, and zero otherwise. B has been then reordered such that Parliamentarians of the same party are located as consecutive rows/columns.

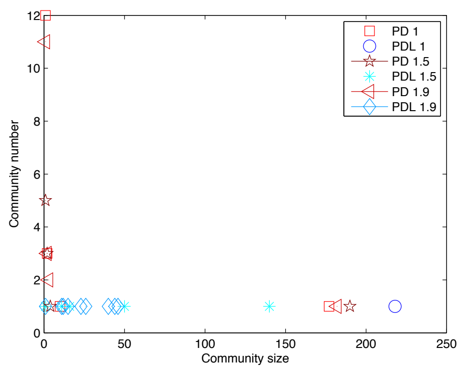
Figure 5.6 shows how the two political parties PDL (rows 304:521) and LN (rows 45:98), that supported the center-right government, progressively reduce their intra-group similarity, while the opposition parties PD (rows 99:303), IDV (rows 24:44), and UDC (rows 546:578) present the opposite trend, i.e. in the first three semesters their intra-group similarity slightly diminishes, in the second three semesters, on the contrary, it increases. It is interesting to note that members of FL (rows 1:23) maintain their high similarity for all the periods, although they separated from PDL in 2010. Another important observation regards the new formed group PT, whose Representatives come from the center-left parties. Although this was constituted in the sixth semester to avoid the government fall, its members showed a good political affinity since the first semester (rows/columns 522:545). The figures clearly show the boosting of agreement from the first to the last semester.

Network representation of Italian Parliament voting records

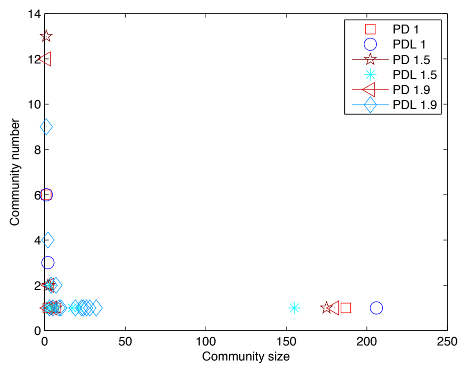
In this section we apply network analysis to the voting records of Italian Parliament to verify if the results obtained with the approaches employed in the previous sections are comparable when changing the analysis method. The binary matrix B, derived from the similarity matrix M, can be used to build an undirected and unweighted network $\mathcal{N}$, where nodes correspond to Parliamentarians and there exists an edge between two



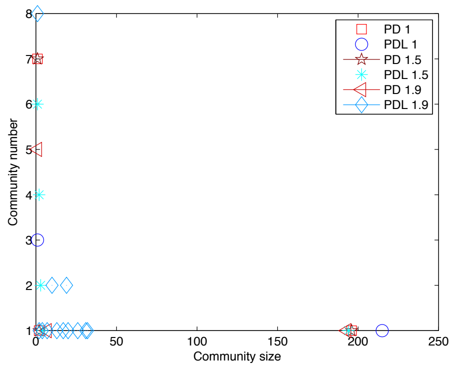
(a) I Semester



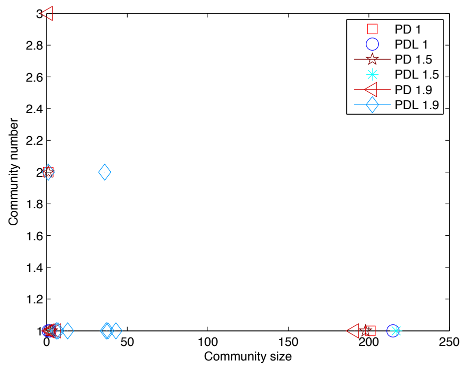
(b) II Semester



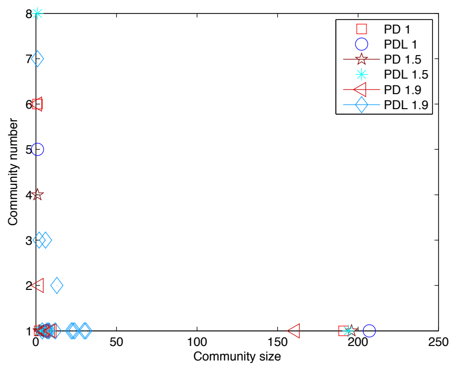
(c) III Semester



(d) IV Semester



(e) V Semester



(f) VI Semester

Fig. 5.8. Number of communities in which the two main parties PDL and PD are split and respective size.

nodes p_i and p_j if the entry B_{ij} is 1. This means that two Representatives are connected if they voted in the same way in at least 60% of the overall roll calls. The community structure of $\mathcal{N}$ can then be investigated by optimizing the well known concept of *modularity* [50]. Remember that it is based on the intuitive idea that a community should have more internal connections among its nodes than interconnections between its nodes and those in other communities. Let's recall that the modularity is defined as

$$Q = \frac{1}{2r}\sum_{ij}(A_{ij} - \gamma\frac{k_i k_j}{2r})\delta(C_i, C_j)$$

where r is the number of edges in the network, k_i is the degree of node i, C_i is the community which i belongs to, and $\delta(C_i, C_j)$ is 1 if nodes i and j belong to the same community, 0 otherwise. γ is a resolution control parameter introduced by Granell et al. [29] to overcome the resolution problem stated in [22] and study community structure at multiple scales. In fact it has been proved that the optimization of modularity has a topological resolution limit that depends on both the total size of the network and the interconnections of groups. This implies that small, tightly connected clusters could not be found. Thus, searching for partitioning of maximum modularity, may lead to solutions in which important structures at small scales are not discovered. When $\gamma = 1$ the equation reduces to the standard formulation of modularity [50].

We used an algorithm optimizing modularity [57] extended with the resolution parameter, and executed the method with three different values of γ: 1, 1.5, 1.9. The latter two values have been chosen to analyze the existence of sub-communities inside those obtained with $\gamma = 1$ that cannot be found by optimizing modularity because of the resolution problem.
Figure 5.7 shows how modularity values vary during the seven semesters for all the three resolution parameters chosen. The figure clearly points out a sharp decrease of modularity in the 6th period and a drastic reduction in the 7th one. In order to better analyze the community structure detected by the algorithm, Figure 5.8 shows the number of communities in which the two main parties PDL and PD have been split. We do not report the results for the other parties because their behavior is analogous to the coalition they belong to. Since the size of the largest community is 218 (i.e. the number of PDL members), the first coordinate varies between 1 and 218. The second coordinate, for each value of γ, reports the number of subgroups of that size obtained by the algorithm. Figure 5.8(a) shows that, with $\gamma = 1$ PDL is grouped in a unique community, while PD is clustered in a big community of 190 members and other 14 members are split in 7 small communities. When $\gamma = 1.5$ the situation is almost the same. However, when $\gamma = 1.9$, PD continues to have a big community of size 192, while PDL is split in 14 communities of size varying between 1 and 46. The rather interesting result is that this behavior is maintained for all the semesters. Thus, while PD remains cohesive for all the semesters, independently of the γ value, PDL is divided in many subgroups since the first semester, when its degree of aggregation was considered very high, and as obtained with the other approaches described in the previous sections.

Thus modularity allows a deeper analysis of the internal agreement of parties and can provide insights of early and unexpected changes a political party could encounter. Moreover, it affords an explicit and clear view of the steady fragmentation of the coalition endorsing the center-right government that culminated in its fall.

5.4.4 The 7th Semester

The analysis described in the previous sections mainly considered the first six semesters. We decided to separate the last semester because the voting behavior of Parliamentarians had an abrupt alteration, as testified also by the results obtained by all the employed methods. First of all, the number of voted measures is less than the fifth part of the other semesters. Furthermore, it happened that the political party organization completely disappeared, and each Parliamentarian voted independently of his group.

Figure 5.9 gives a clear representation of this situation. In fact, the application of SVD on this semester (Figure 5.9(a)) shows a polarization of all the parties on the first coordinate, and distinguishes between center-left and center-right only on the bipartisan coordinate. Hierarchical clustering returns a unique cluster including all the parties (Figure 5.9(b)), and the visualization of the voting matrices (Figure 5.9(c)) depicts high fragmentation. Finally, Figure 5.9(d) shows that modularity optimization with $\gamma = 1$ extracts a group of 156 and another of 19 members from PD, and two groups of 94 and 52 members from PDL. However these groups are clustered together, thus confirming the results of the other approaches. For higher values of γ, both parties are split in small groups of at most 20 Parliamentarians, and the communities found are constituted by members of almost all the political parties.

It is worth noting that, as already pointed out, Figure 5.7 indicates an abrupt lowering of modularity value in the 7th semester that explains the loss of community structure.

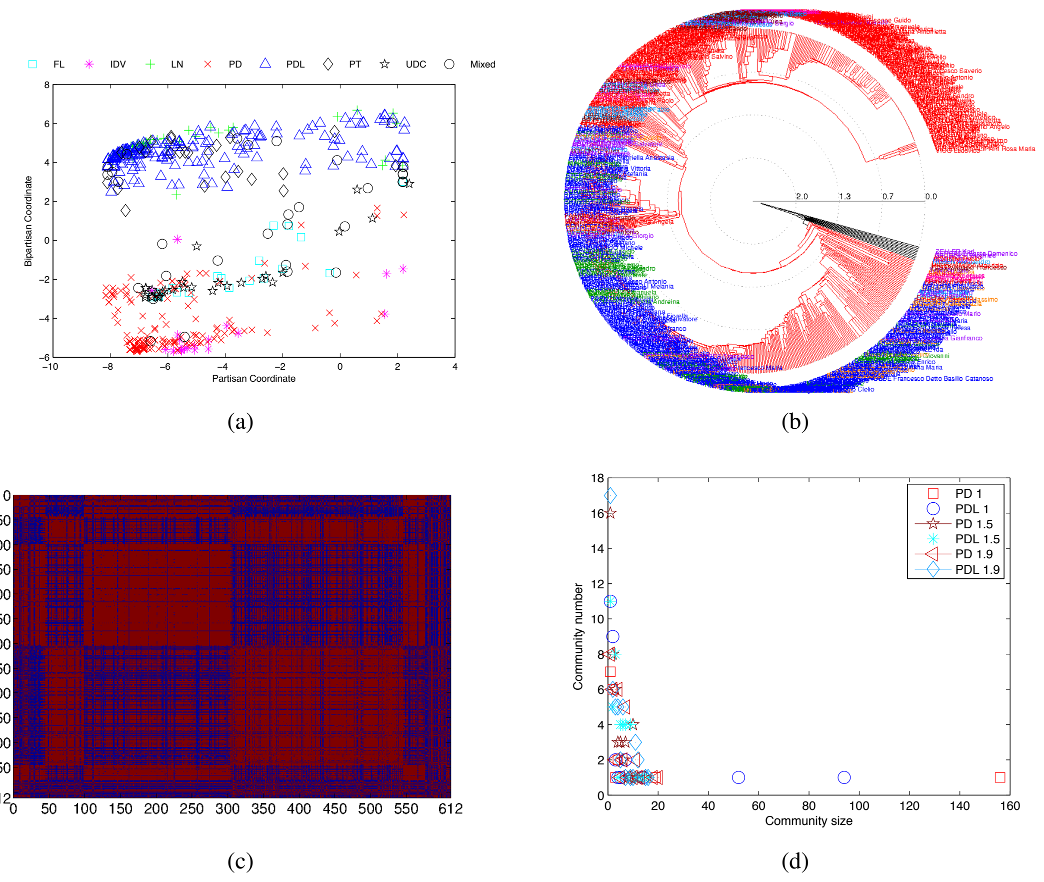


Fig. 5.9. Results obtained by applying SVD (a), hierarchical clustering (b), visualization of the similarity matrix (c), and community detection (d) on the 7th semester.

6 Conclusion to Part II

Part II of this thesis introduced a new approach for extracting patterns from political data.

In particular, it proposed an investigation of the voting behavior of Italian Parliament in the last years to find voting patterns by employing different computational tools. Though studies of this kind exist for different political institutions from US and Europe, as far as we know, this is the first tentative of exploring Italian Parliament with data mining and network analysis methods. Networks among the Parliamentarians have been generated at consecutive time periods and community structure investigated at multiple scales. By delving the voting records of Representatives, organizational structure of Parliament has been characterized, and latent information discovered. All the used methods showed to be effective at identifying political patterns, and at providing insights on the temporal evolution patterns of groups and their cohesiveness.

7

General conclusion and future work

This thesis presented some results in the field of Information Extraction, where patterns define the information to extract and extraction procedures realize how these patterns can be effectively discovered from data. Because information extraction is a relevant topic in multiple communities characterized by different kinds of data sources, such as image processing and political science, patterns are strongly dependent on the kind of input data which they must be detected from and appropriate extraction procedures must be adopted for this purpose. In particular, patterns in image processing, which are pixels regions or objects, can be discovered by image segmentation and edge detection techniques. In addition, image compression algorithms could require the extraction of patterns which will characterize the compression phase. In political science, network analysis and data mining are able to detect patterns like trends, voting profiles and interactions among political subjects. Our proposed techniques broaden in these two directions, by proposing new image segmentation and compression techniques and a new approach analyzing political data. In particular, we proposed an image segmentation technique, which extends the well-known *Normalized Cut* concept, normally needing the number of partitions. By adopting a genetic approach, our technique is able to partition the input image in an arbitrary number of partitions which is dynamically computed during the algorithm execution. In image compression, we extended the traditional Lempel-Ziv techniques in 2D, by introducing a parser extracting maximal and irredundant ”not solid” patterns from images in order to improve the compression gain. In political science, we found automatic tools for characterizing the modifications and changes occurred inside the Italian Parliament during the XVI legislature, by using data mining and network analysis techniques.

Future work will continue in the direction of pattern extraction from images and political data. In some contexts, color is a feature of prior importance for characterizing object patterns inside the images. Furthermore, particularly complex images, such as medical and satellite images, are composed of textures whose detection is relevant for discriminating different region patterns. Consequently, we will focus on the segmentation process to partition more complex images, by combining the contour information considered in our approach with color and texture features of the regions. Thus we expect a further improvement of our approach in the segmentation accuracy. In politics, it is sometimes difficult to discover for some subjects their membership to a given group. This is mainly because their ideology is not completely clear in the political context and traditional community detection methods tend to force, for their nature, this hard membership to a single group. Consequently, future work aims at applying overlapping community detection methods to better uncover hidden collaborations among Parliamentarians of different political membership.

8

Publications underlying this thesis

Alessia Amelio and Clara Pizzuti. An Evolutionary and Graph-based method for Image Segmentation. *In Proceedings of 12th International Conference on Parallel Problem Solving from Nature (PPSN 2012), Taormina, Italy, 2012.*

Alessia Amelio, Alberto Apostolico and Simona E. Rombo. Image Compression by 2D Motif Basis. *In Proceedings of IEEE Data Compression Conference (DCC 2011), Snowbird, UT, USA, 2011.*

Alessia Amelio and Clara Pizzuti. Analyzing Voting Behavior in Italian Parliament: Group Cohesion and Evolution. *In Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), Istanbul, Turkey, 2012.*

References

1. A. Apostolico and S. Lonardi. Compression of biological sequences by greedy off-line textual substitution. In *Data Compression Conference*, pages 143–152, 2000.
2. A. Apostolico, L. Parida, and S. E. Rombo. Motif patterns in 2D. *Theoretical Computer Science*, 390(1):40–55, 2008.
3. S. H. Bae. Information retrieval via universal source coding. *Ph.D. dissertation, Georgia Institute of Technology*, 2008.
4. U. C. Benz, P. Hofmann, G. Willhauck, I. Lingenfelder, and M. Heynen. Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58(3-4):239–258, 2004.
5. A. P. Berg and W. B. Mikhael. A survey of mixed transform techniques for speech and image coding. In *ISCAS (4)*, pages 106–109. IEEE, 1999.
6. S. Beucher. The watershed transformation applied to image segmentation. *Scanning Microscopy Supplement*, 6:299–314, 1992.
7. C. A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions on Image Processing*, 3(2):162–177, 1994.
8. T. J. Brazill and B. Grofman. Factor analysis versus multi-dimensional scaling: binary choice roll-call voting and the us supreme court. *Social Networks*, 24(3):201 – 229, 2002.
9. A. P. Carleer, O. Debeir, and E. Wolff. Assessment of very high spatial resolution satellite image segmentations. *Photogrammetric Engineering and Remote Sensing*, 71(11):1285–1294, 2005.
10. A. Cavallaro, E. D. Gelasca, and T. Ebrahimi. Objective evaluation of segmentation quality using spatio-temporal context. In *ICIP (3)*, pages 301–304, 2002.
11. C. W. Chen, J. Luo, and K. J. Parker. Image segmentation via adaptive k-means clustering and knowledge-based morphological operations with biomedical applications. *IEEE Transactions on Image Processing*, 7(12):1673–1683, 1998.
12. S. Chen, J. Luo, Z. Shen, X. Hu, and L. Gao. Segmentation of multi-spectral satellite images based on watershed algorithm. In *Proceedings of International Symposium on Knowledge Acquisition and Modeling*, pages 684–688, 2008.
13. S. Chen and K. Zhang. Robust image segmentation using fcm with spatial constrains based on a new kernel-induced distance measure. *IEEE Transactions on Systems Man and Cybernetics B*, 34:1907–1916, 2004.
14. Z. Chen, Z. Zhao, P. Gong, and B. Zeng. A new process for the segmentation of high resolution remote sensing imagery. *International Journal of Remote Sensing*, 27(22):4991–5001, 2006.
15. P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray. Using vector quantization for image processing. In *Proc. IEEE*, pages 1326–1341, 1993.
16. T. Cour, F. Bénézit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005))*, pages 1124–1131, 2005.
17. V. Dey, Y. Zhang, and M. Zhong. A review on image segmentation techniques with remote sensing perspectives. In *Proceedings of ISPRS TC VII Symposium*, volume XXXVIII, pages 31–42, 2010.
18. R. O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
19. P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.
20. Y. Fisher. Fractal image compression. Technical Report 12, Department of Mathematics, Technion Israel Institute of Technology, 1992. SIGGRAPH '92 COURSE NOTES.
21. D. B. Fogel. What is evolutionary computation? *IEEE Spectr.*, 37(2):26–32, February 2000.

22. S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. National Academy of Science, USA*, 104(36), 2007.
23. P. Fränti, O. Nevalainen, and T. Kaukoranta. *Compression of Digital Images by Block Truncation Coding: A Survey*. Research reports / Computer Science, University of Turku. University of Turku, 1993.
24. V. Di Gesú and G. Bosco. Image segmentation based on genetic algorithms combination. In Fabio Roli and Sergio Vitulano, editors, *Image Analysis and Processing (ICIAP 2005)*, volume 3617 of *Lecture Notes in Computer Science*, pages 352–359. Springer Berlin / Heidelberg, 2005.
25. R. Giancarlo and R. Grossi. Suffix tree data structures for matrices. In A. Apostolico and Z. Galil, editors, *Pattern matching algorithms*, chapter 10, pages 293–340. Oxford University Press, 1997.
26. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
27. S. Golomb. Run-length encodings. *Information Theory, IEEE Transactions on*, 12(3):399–401, 1966.
28. R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
29. C. Granell, S. Gómez, and A. Arenas. Unsupervised clustering analisys : A multiscale complex network approach. *Journal of Bifurcation and Chaos, in press*, 2012.
30. A. Halder and N. Pathak. An evolutionary dynamic clustering based colour image segmentation. *International Journal of Image Processing*, 4:549–556, 2011.
31. J. D. Helterbrand. One pixel-wide closed boundary identification. *IEEE Transactions on Image Processing*, 5(5):780–783, 1996.
32. S. Hix, A. Noury, and G. Roland. Power to the parties: Cohesion and competition in the European Parliament. *British Journal of Political Science*, 35(2), 2005.
33. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
34. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.
35. A.K. Jain and R. Cg. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
36. A. Jakulin and W. Buntine. Analyzing the US Senate in 2003: Similarities, networks, clusters and blocs. *http://eprints.fri.uni-lj.si/archive/00000146/*, 2004.
37. L. Jiao. Evolutionary-based image segmentation methods. *Image Segmentation*, 2011.
38. L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley series in probability and mathematical statistics. Applied probability and statistics. Wiley, 2005.
39. G. Kuntimad and H. S. Ranganath. Perfect image segmentation using pulse coupled neural networks. *IEEE Transactions on Neural Networks*, 10(3):591–598, 1999.
40. C.C. Lai and C.Y. Chang. A hierarchical evolutionary algorithm for automatic medical image segmentation. *Expert Syst. Appl.*, 36(1):248–259, January 2009.
41. B. J. Lei, A. Hendriks, and M. J. T. Reinders. On feature extraction from images, June 1999.
42. A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22:75–81, 1976.
43. A. Lempel and J. Ziv. Compression of two-dimensional data. *IEEE Transactions on Information Theory*, 32(1):2–8, 1986.
44. T. K. Leung and J. Malik. Contour continuity in region based image segmentation. In *5th European Conference on Computer Vision, ECCV'98*, pages 544–559, 1998.
45. H.T. Li, H.Y. Gu, Y.S. Han, and J.H. Yang. An efficient multi-scale segmentation for high-resolution remote sensing imagery based on statistical region merging and minimum heterogeneity rule. In *Proceedings of International Workshop on Earth Observation and Remote Sensing Applications*, pages 1 – 6, 2008.
46. L. Li, J. Ma, and Q. Wen. Parallel fine spatial resolution satellite sensor image segmentation based on an improved pulse-coupled neural network. *International Journal of Remote Sensing*, 28(18):4191–4198, 2007.
47. K. T. Macon, P. J. Mucha, and M. A. Porter. Community structure in United Nations General Assembly. *Physica A: Statistical Mechanics and its Applications*, 391(1-2):343–361, January 2012.
48. T. Maxwell and Y. Zhang. A fuzzy logic approach to optimization of segmentation of object-oriented classification. In *Proceedings of SPIE 50th Annual Meeting - Optics and Photonics*, pages 1–11, 2005.
49. M. Merzougui, A. EL. Allaoui, M. Nasri, M. EL. Hitmy, and H. Ouariachi. Evolutionary image segmentation by pixel classification and the evolutionary Xie and Beni criterion - application to quality control. *International Journal of Computational Intelligence and Information Security*, 2(8):4 – 13, 2011.
50. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E69:026113, 2004.
51. A. Pajala, A. Jakulin, and W. Buntine. Parliamentary group and individual voting behavior in Finnish Parliament in year 2003 : A group cohesion and voting similarity analysis. *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.2295*, 2004.

52. N. R. Pal and S. K. Pal. Entropy: a new definition and its applications. *IEEE Transactions on System, Man and Cybernetics*, 21(5):1260–1270, 1991.
53. S. K. Pal, A. Ghosh, and B.U. Shankar. Segmentation of remotely sensed images with fuzzy thresholding, and quantitative evaluation. *International Journal of Remote Sensing*, 21(11):2269–2300, 2000.
54. T. N. Pappas. An adaptive clustering algorithms for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, 1992.
55. Y.J. Park and M.S. Song. A genetic algorithm for clustering problems. In *Proc. of 3rd Annual Conference on Genetic Algorithms*, pages 2–9, 1989.
56. M. Paulinas and A. Uinskas. A survey of genetic algorithms applications for image enhancement and segmentation. *Information Technology And Control, Kaunas, Technologija*, 36(3):278 – 284, 2007.
57. C. Pizzuti. Boosting the detection of modular community structure with genetic algorithms and local search. In *Proc. of the 27th Symposium on Applied Computing (SAC 2012)*, pages –, 2012.
58. M. A. Porter, P.J. Mucha, M. E. J. Newman, and A. J. Friend. Community structure in the United States House of Representatives. *Physica A: Statistical Mechanics and its Applications*, 386(1):414–438, December 2007.
59. S. E. Rombo. Optimal extraction of motif patterns in 2D. *Information Processing Letters*, 109(17):1015–1020, 2009.
60. P. K. Sahoo, S. Soltani, and A. K. C. Wong. A survey of thresholding techniques. *CVGIP*, 41:233–260, 1988.
61. P. K. Sahoo, S. Soltani, A. K.C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41(2):233–260, 1988.
62. A. Sarkar, M. K. Biswascand, and K. M. Sharma. A simple unsupervised MRF model based image segmentation approach. *IEEE Transactions on Image Processing*, 9(5):801–812, 2000.
63. A. Shahbahrami, R. Bahrampour, M. S. Rostami, and M. A. Mobarhan. Evaluation of huffman and arithmetic algorithms for multimedia compression standards. *CoRR*, abs/1109.0216, 2011.
64. B. U. Shankar. Transactions on rough sets vii. chapter Novel classification and segmentation techniques with application to remotely sensed images, pages 295–380. Springer-Verlag, Berlin, Heidelberg, 2007.
65. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
66. D. Sinha and E.R. Dougherty. *Introduction to Computer-Based Imaging Systems*. Tutorial texts in optical engineering. SPIE Optical Engineering Press, 1998.
67. G. Strang. *Linear Algebra and its Applications*. IV edition, Thomson Brooks/Cole, 2005.
68. P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson International Edition, 2006.
69. S. G. Tzafestas and S. N. Raptis. Image segmentation via iterative fuzzy clustering based on local space-frequency multi-feature coherence criteria. *Journal of Intelligent and Robotic Systems*, 28(1-2):21–37, 2000.
70. R. Urquhart. Graph theoretical clustering based on limited neighborhood sets. *Pattern Recognition*, 15(3):173–187, 1982.
71. A. Visa, K. Valkealahti, and O. Simula. Cloud detection based on texture segmentation by neural network methods. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1001–1006, 1991.
72. J. Vrindavanam, S. Chandran, and G. K. Mahanti. Article: A survey of image compression methods. *IJCA Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET 2012)*, icwet(1):12–17, March 2012. Published by Foundation of Computer Science, New York, USA.
73. A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter. Party polarization in Congress: A network science approach. *http://arxiv.org/abs/0907.3509*, 2009.
74. M. J. Weinberger, J. Ziv, and A. Lempel. On the optimal asymptotic performance of universal ordering and discrimination of individual sequences. In *Data Compression Conference*, pages 239–246, 1991.
75. T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, June 1984.
76. Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and applications to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.
77. Y. Xu, V. Olman, and E. C. Uberbacher. A segmentation algorithm for noisy images: Design and evaluation. *Pattern Recognition Letters*, 19:1213–1224, 1998.
78. Y. Yang, C. Han, and D. Han. A Markov random field model-based fusion approach to segmentation of SAR and optical images. In *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 802–805, 2008.
79. C. T. Zahn. Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.
80. Y. Zhang, A.J. Friend, A. L. Traud, M. A. Porter, J. H. Fowler, and P. J. Mucha. Community structure in Congressional cosponsorship networks. *Physica A: Statistical Mechanics and its Applications*, 387(7):1705–1712, March 2008.
81. C. Zhong, Z. Zhongmin, Y. DongMei, and C. Renxi. Multi-scale segmentation of the high resolution remote sensing image. In *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3682–3684, 2005.
82. L. Zhu. Computational political science literature survey, August 2010.

83. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
84. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.