Spatial clustering of structured objects

Antonio Varlaro

Dipartimento di Informatica
Università degli Studi di Bari
varlaro@di.uniba.it

Promotor: Prof. Donato Malerba

A dissertation submitted in partial satisfaction of the requirements
for the degree of
Doctor of Philosophy in Computer Science
in the Graduate Division of the
University of Bari, Italy

Credits

This dissertation was typeset using these shareware programs:

- TeXnicCenter 1 Beta 6.21
  available at: `http://www.texniccenter.org/`
- MikTEX 2.1
  available at: `http://www.miktex.de`

Thesis Supervisor

Prof. Donato Malerba

Chairperson of the Supervisory Committee

Member of the Supervisory Committee

Member of the Supervisory Committee

Submitted *February 2008*

Contents

List of Figures

List of Tables

List of Symbols

$\#A$	Cardinality of the generic set A, page 19
$\mathcal{B}$	Herbrand base, page 53
$\mathcal{H}$	Herbrand universe, page 53
$\mathcal{I}$	Interpretation, page 53
$\mathcal{L}$	First-order logic language, page 52
$\mathcal{T}$	Logical theory, page 54
$\models$	Logical entailment, page 53
$\prec$	Generality order over clauses, page 55
θ	Substitution, page 53
$\vdash$	Logical derivation, page 54
$APTS$	Air Pollution Testing Station, page 131
BK	Background Knowledge, page 49
DB	The set of objects in the database to be considered for clustering, page 19
$DBMS$	DataBase Management System, page 44
DM	Data Mining, page 6
E	Set of training examples, page 55
E^+	Set of positive training examples, page 55
E^-	Set of negative training examples, page 55
GIS	Geographic Information System, page 15
ILP	Inductive Logic Programming, page 51
k	Number of detected cluster, page 9
KDD	Knowledge Discovery in Database, page 5

lgg	Least General Generalization, page 50
LHM	Least Herbrand Model, page 54
$MRDM$	Multi-Relational Data Mining, page 10
$MRDM$	Multi-Relational Data Mining, page 46
n	Number of objects (or observations) in the database, page 20
RDM	Relational Data Mining, page 46
SNA	Social Network Analysis, page 108

Acknowledgments

Bryant H. McGill said:

> *"Knowledge is that possession that no misfortune can destroy, no authority can revoke, and no enemy can control. This makes knowledge the greatest of all freedoms."*

Reading this statement and pondering on its meaning I can not help expressing my gratitude to my supervisor, Prof. Donato Malerba. He conveyed to me his passion for research and dedication in work. His example and guidance have been the main enablers of this work. His teachings and suggestions have been a lamp that drove me in the unexplored and tortuous paths of knowledge.

I can not help thanking Annalisa Appice that has always provided me with her helpful advices and valuable suggestions. She has been for me a smart colleague and a model to imitate, and still she is.

Many thanks to Michelangelo Ceci, Costantina Caruso, Margherita Berardi and Antonio Turi that, besides being phenomenal colleagues, are dear friends.

A special word to enjoyable company offered by Corrado Loglisci and Oronzo Altamura that have always been for me dear and true friends. They have never forsaken me, especially in need, and have alleviated my thoughtful or discouraged moments with their jokes and sweets.

I would like to thank Prof. Floriana Esposito, for her useful suggestions and for the opportunity she gave me to work in LACAM (Knowledge Acquisition & Machine Learning Lab).

I would like to thank people working (and living) in LACAM and, particularly, Claudia D' Amato, Luigi Iannone, Mimmo Redavid and Ignazio Palmisano for sharing with me long days of hard work, but also to Maria Teresa Basile, Paolo Buono, Tatiana Cassio, Bruna & Marco Degemmis, Nicola Di Mauro, Nicola Fanizzi, Stefano Ferilli and Pasquale Lops. They share offices close to that one where I worked and countless meet me during each day.

Thanks to Lorenzo Angiuli, Adriana Primicino and Maria Camarrota, working at ARPA Puglia (Bari) for their kind support in the gathering of data concerning the air pollution in Apulia region. Special thanks are reserved for Giampiero Rosa, Biagio Di Napoli and all the people working at Exhicon s.r.l., a small but smart firm that always minds the innovation and makes research one of its primary goal.

Still, I would like to thank friends, my constant and patient supporters that have never refused to accept my binding commitments even when work overlapped

free time planned to be spent with them. More of them, a special word is dedicated to my girlfriend, Anna, for tolerating me with endless patience and encouraging my efforts.

Finally but not least, the most grateful thanks are dedicated to my parents and my sister, Cinzia, because they have always accompanied my smiles with their charge of enthusiasm and wiped my tears with their experienced and precious suggestions. This work and all my successes are due to my parents' countless sacrifices, their encouragements, the boundless confidence they placed in me.

Abstract

The huge amount of available data collected automatically or manually and the limited capacity to elaborate it by humans make the definition and enhancement of automated analysis procedures a necessity.

Data Mining (DM) is the answer to this demand: it is defined as the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data and describing them in a concise and meaningful way. As a research field, DM stems from different disciplines such as artificial intelligence (more specifically, machine learning), database technology, statistics and pattern recognition.

A prominent example of DM task which has been investigated in several disciplines is *clustering*. It is a descriptive task which aims at identifying natural groups (or *clusters*) in data by relying on a given criterion that estimates how two or more objects are similar each other. The goal is to find clusters of objects (observations on provided data) such that their similarity is high among objects in the same cluster and is low among objects belonging to different clusters. Since no information is provided in advance about both the number of clusters to be mined and the correct membership of data with respect to clusters, such form of mining is said to be *unsupervised*.

Different clustering methods have been reported in the literature. They mainly differ for the criteria used to group the data and the type of data they can manage.

As to the criteria, two classes of clustering algorithms are of interest in this work: *conceptual clustering* and *graph-based partitioning*. Conceptual clustering methods consider each cluster as a concept and partition training observations on the basis of the symbolic descriptions which can be associated with each cluster, while graph-based partitioning methods consider observations as nodes of a graph whose arcs indicates relations among observations, and cluster observations by partition the graph according to nodes connectivity and other graph structural characteristics.

As to the data, several clustering algorithms assume that observations are represented as tuples of a single database relation. This "single table assumption" prevents the consideration of relationships between observations as well as the analysis of data which are logically modelled through several database tables. To overcome these limitations, many researchers have recently started investigating the *(Multi-)Relational Data Mining* (MRDM) approach to data analysis. In some sense, graph-based partitioning methods also represent a particular class of (multi-)relational clustering algorithms.

In this dissertation, we focus our attention on *spatial clustering*, that is, clustering data characterized by a spatial dimension. Spatial data have a specific shape and position in a given reference frame which implicitly define spatial relationships (e.g., intersection, adjacency, and so on). Moreover spatial data represent objects of different types (e.g., towns and rivers) which are naturally described by different relations of a relational database. For this reason, the most natural approach to spatial clustering seems to be the multi-relational one. We consider training observations as complex units of analysis described by a number of database relations, moreover we see training observations as nodes of a graph and we view at clustering as a graph-based partitioning. The result of the clustering is a logical theory which describes each partition (or cluster) as in the case of conceptual clustering.

The thesis is organized as follows. First, we analyse different approaches to clustering spatial data, we present some solutions devised to manipulate such data and we illustrate the main strengths and drawbacks of each method. Then, we present a new method for clustering complex training observations arranged in a graph (called *discrete spatial structure*). The method, named CORSO (*Clustering Of Related Structured Objects*), clusters observations on the basis of two criteria: i) each cluster should correspond to a connected subgraph of the original discrete spatial structure, and ii) observations in a cluster should be similar according to a similarity measure defined for structured objects (i.e., objects described by multiple database relations). Finally, we present specific issues of the proposed method, namely the selection of the seed observations and the evaluation of the homogeneity of a cluster.

Some experimental results are reported. They show the application of CORSO to both artificial and real data concerning different application domains. Moreover, they contribute to illustrating the limits and the potentialities of CORSO and provide empirical evidence to theoretical analysis.

Chapter 1

Introduction

Last decades have been characterized by a huge amount of available data coming from different human activities. Most of them are often left unused due to the limited capacity to be elaborated by humans. This happens in spite of the common need for turning such data into useful knowledge. The information and knowledge gathered can be profitably used for applications ranging from business management, market analysis, science exploration, social analysis, geographical studies, medical diagnoses and so on, and every knowledge nugget gained is an additional chance to make progress in each of these areas. The need for extracting knowledge from raw data is undergoing a growth increasingly substantial and fast as the availability of more and more sophisticated and powerful technological tools widen the gap between data produced and those analysed. Hence, the definition and enhancement of automated analysis techniques represent a necessity by now.

In the attempt to fill this gap, an evolutionary path has characterized the database industry as concerns the development of the following functionalities: data collection and database creation, data management, data analysis and understanding [HK01]. The last step of this path has given rise to the field of *Knowledge Discovery in Database* (KDD), defined as the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data and describing them in a concise and meaningful way [FSS96]. Such a process is interactive and iterative and involves numerous steps, as depicted in Figure 1.1. These steps include:

- *Data cleaning*, that aims at removing noises and inconsistencies in data
- *Data integration*, where data coming from different sources are gathered and merged together
- *Data selection*, data relevant to the analysis task are extracted by the great deal of available data
- *Data transformation*, data are transformed from the format in which they are available to a format suitable to be analysed

- *Data mining*, consisting in the application of specific algorithms for extracting patterns from data
- *Pattern evaluation*, that aims at identifying truly interesting patterns from the mined ones according to some interestingness measures
- *Knowledge presentation*, where visualization and knowledge representation techniques are adopted to present the mined knowledge in a expressive and user-friendly way

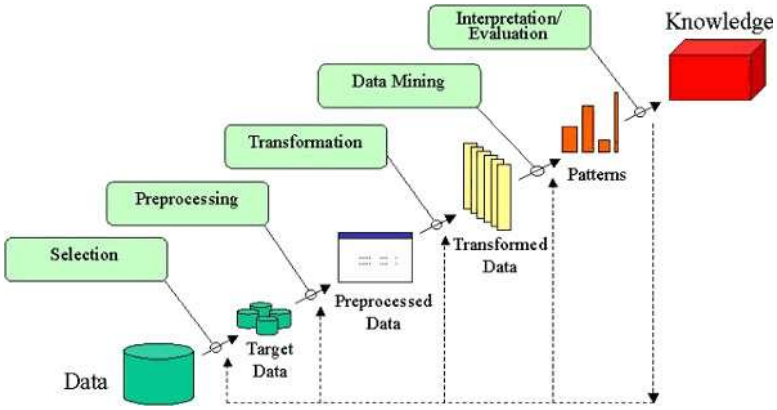


FIGURE 1.1: The KDD process

For instance, let us suppose that a chain of computer shops want to have suggestions about the arrangement of goods in its points of sale. A KDD system could gather data from the data sources held by each branch into a singular database (data integration), clean data from noise or missing information by removing records having null or invalid values (data cleaning), identifying tables describing commercial transasction and cutting out unecessary tables (data selection) and tranforming data into a attribute-value formalism (data transformation). After that, the system could perform an association rules discovery task to detect relations between items sold (data mining step). After a set of association rules have been discovered, the system could highlight those ones with high support/confidence values (pattern evaluation) and present them to end-user by means of charts (knowledge presentation). In this way, managers and market analysts could easily discover that, for instance, 65% of people buying a printer also buys a card-reader (because they intend to use printer mainly to print their digital photos), so having the suggestion of placing the card-readers close to printers shelf.

According to the above observations, it is clear that Data Mining is the most important step in the process of generating novel knowledge from raw data to the point that the term *data mining* is often used as a synonym for KDD.

Data Mining (DM) can be described as the step of KDD process consisting of applying computational techniques that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (or models) over the data [FSS96].

Results obtained from DM applications can be used either as input for automated decision support systems or as models to be validated by domain experts. For instance, managers of business companies could profitably use DM methods to discover fraudulent cases or doctors could consult DM systems to investigate on novel medical cause-effect relationships.

Numerous data mining methods exist, including *predictive* data mining methods which typically result in models that can be used for prediction and classification, and *descriptive* data mining methods which can be used for exploratory data analysis to discover individual patterns, such as associations, clusters, and other patterns that can be of interest to the user [MLBM03].

One of the fundamental tasks in DM is *Clustering*. Starting from a set of unlabelled observations (or *objects*), the purpose is to detect groups of objects (or *clusters*) such that each object results to be similar to the objects belonging to the same cluster and dissimilar with respect to objects belonging to different clusters.

Clustering is a form of *unsupervised learning*, that is, neither the number of classes to be learned nor the membership of training examples to any of target classes are known in advance, in contrast with the *supervised learning*, that relies on information about the class which each training example belongs to.

Because of the challenging goal of clustering and the usefulness of mined patterns, this data mining task can be profitably applied to a great deal of fields: in astronomy clustering methods can be applied to automatically group stars into galaxies, in military field they can be used to detect clusters of mines, in business contexts customers can be grouped into classes associating them with a group profile and in geography clustering can be adopted to characterize areas on the basis of geomorphological features, to cite some examples.

Furthermore, recent years have been characterized by an increasing attention to clustering approaches specifically devoted to deal with spatial data. *Spatial data* differ from non-spatial data in that they also have spatial attributes, that is, attributes saying something about the space occupied (position in a reference frame, size and shape, for instance) in addition to classical descriptive attributes. This framework poses additional issues to be faced: taking into account spatial features in data also requires to consider relations that are implicitly defined for the observations (or *objects*), such as distance, direction and topological relations. In addition, the attributes of a spatial objects may be affected by the attributes of nearby objects.

For instance, a spatial environment could be represented by cities in a country. They can be considered spatial objects since a position in a reference frame (e.g., latitude and longitude) and a geometrical shape (point, line, polygon or polyhedron) can be associated with each of them. Here, relations such as distance or reachability implicitly exist over data and neighbouring places can affect some attributes of a given city. For instance, the presence of a steel factory can alter air pollution measurements in nearby urban areas. In such domain, spatial clustering could be applied to automatically detect regions of nearby similar cities in order to support public authority to take decisions.

Hence, it is clear that clustering could be of great importance in practically every

discipline. The natural question is: which aspects should be taken into account in spatial clustering? Desirable properties for a spatial clustering system are [Kol01, Ber02]:

- few, or even no parameters (or other a-priori knowledge about the target clusters) have to be required for the algorithm
- ability to deal with both categorical and numerical attributes
- scalability in time consumed, that is, performance decay should follow the same trend of the size of input data
- ability to identify irregular or nested shapes, in order to better model real arrangements of data
- noise tolerance, that is, performances in clusters detection should not decay in presence of noise in data
- insensitivity to the order of input, since clustering results should be insensitive to the order in which data are read
- ability to work with high dimensional data, that is, data with a large number of descriptive features
- interpretability of results

All aspects presented till now stress that spatial clustering is a challenging learning task that can provide answers to non-trivial problems in many disciplines, but at the same time requiring a deep and precise analysis of issues related to this type of mining.

In this thesis we investigate the main issues concerning the clustering of spatial data. In particular, we discuss strengths and weaknesses of well-known spatial clustering methods available in literature and we propose a novel method that combines features characterizing different approaches to overcome limitations of existing algorithms. Finally, we evaluate the proposed method on both artificial and real datasets and concluding remarks and future works close this dissertation.

1.1 Motivations and Contribution

The problem of clustering spatial data has been extensively investigated in literature as witnessed by a large number of surveys on this topic [EKSX98, HKT01, Kol01, Ber02]. However, much of the proposed methods have been developed to deal with data described in terms of points while only few works have been designed to detect spatial clusters of areal data. For this reason, usually spatial clustering methods consider input data described in terms of observations associated with an exact location in a reference frame (or *geo-referenced*, in case of geographical environments). Here, an observation is represented as a triple $\langle (x_i, y_i), z_i \rangle$, such that (x_i, y_i) denotes the location of the point i with respect to some coordinate system, and z_i is the set of measured attributes observed at site i. This simplified

description has characterized early studies in spatial clustering, leading to some widely known methods. One of them is K-MEANS [Mac67] that uses the cluster centre (having the feature values equal to the mean values of the objects belonging to the cluster) as cluster representative. A severe drawback of this method is its high sensitivity to noise: since a cluster is represented by its centre and this is calculated as the average value of the features characterizing the objects in the cluster, detected clusters are heavily influenced by wrong values in case of data are affected by noise. In addition, clusters are represented by a fictitious object (the calculated cluster centre) that rarely represents an useful information for the data analyst. In order to mitigate such drawbacks, K-MEDOIDS [KR90] method was developed that works similarly to K-MEANS, but uses the most centrally located object in the cluster (instead of cluster centre) as its representative.

Nevertheless, both the above methods suffer from another considerable drawback: they find a partition of the dataset where the number k of cluster to be detected must be specified as input parameter. This represents a heavy limitation, since usually the number of clusters in which data are grouped is not known in advance and, generally, this number itself represents a nugget of information to be mined. On the other hand, fixing the number of clusters represents a severe bias that affects, and sometimes prevents, the extraction of useful knowledge from data. In addition, both K-MEANS and K-MEDOIDS can only discover clusters of convex shape.

A method well-known in literature that overcomes such limitation is DBSCAN [EKSX96] that, relying on a density-based approach, finds clusters of arbitrary shape and can handle noise being able to separate dense regions of objects (representing clusters) from noisy data. However, DBSCAN still considers data to be clustered as points arranged in a reference frame, whereas real domains usually deal with space in terms of areas (corresponding to either irregular shapes or regular cells in a grid) and not points. To this aim, a development of this method has been GDBSCAN [SEKX98] that, generalizing the concepts of *neighborhood* and *cardinality*, becomes able to cluster spatially extended objects according to both their spatial and non-spatial attributes. In particular, GDBSCAN extends the definition of neighorhood of an object by allowing to consider each reflexive and symmetric relation (e.g., overlap or 'in contact with') other than Euclidean distance relation. In this way, depending on the considered relation, we are implicitly assuming the existence of a *spatial structure* imposed on data expressing both spatial objects and relations among them.

Such structure can be profitably modelled by means of a graph, known in literature as *neighborhood graph* or *proximity graph* [Tou91, EFKS00, EKS01, And01], where objects are represented by nodes while edges represent links between them. This readable and powerful representation of data, extensively used in pattern recognition [HV03] and supervised learning [GHC01, HC03], has given rise to new spatial clustering techniques [EL01]. Some examples of methods stemming from the combination of clustering and graph-based learning are SUBDUE [JCH01], HPGCL [And03] and CLARANS [NH94]. However, methods belonging to this approach aim at finding regularities in the graph (such as frequently occurring substructures)

or focus only on nodes connectivity (cluster similarity is considered according to the number of links between nodes) ignoring the descriptive features of the objects.

Even if we consider spatial objects (or nodes in graphs) to represent areas instead of points, the approaches mentioned above are not able by themselves to "look inside the objects". Indeed, areal data described in terms of some areal representative, such as centroid [Vis83] or polygon, is still too restrictive since observations for an area are descriptive of one or more primary units, generally of different types, collected within the same area boundary. In this case, data do not only describe features concerning the entire area but also include both attributes that relate to either primary units (e.g., number of persons in a family) or entire areas (e.g., unemployment rate in a urban area) and attributes that refer to relations between primary units (e.g., contact frequencies between households) and between areal units (e.g., migration rate of a city). It is clear that methods neglecting such considerations suffer from severe limitations due to the *single table assumption*: they assume that all data are stored in a single table (or *relation*, in database terminology) and there is one row (or *tuple*) in this table for each object of interest [Wro01]. In fact, considering all data in a spatial object to be descriptive of the entire area corresponds to assume that all information enclosed in the object description can be represented in a single table, having a column (or field) for each considered attribute and a row for each observation. Moving towards a structured concept of spatial object forces to arrange data in different tables, related each others, where each table describes either features corresponding to entire areas or primary units or relations between them.

This data organization is typically adopted by *Multi-Relational Data Mining* (MRDM) that overcomes the limitations of single-table assumption by considering observations arranged in multiple tables belonging to a relational database. Here, the objective is to mine interesting patterns scattered over one or more tables. Such patterns result to be more powerful with respect to those ones mined in general environments, because they are also able to easily model links among heterogeneous data. In order to achieve this goal, MRDM resorts to *Inductive Logic Programming* (ILP) techniques, that adopts the first-order logic formalism (or some subset of it) to efficiently represent both data and relations. In addition, in such environment the same language is adopted to describe both training data and mined models, being able to easily recombine data and models in order to further explore additional hypotheses.

Generally, MRDM works in *learning from interpretation* setting, where each example is a Herbrand interpretation, that is a set of ground facts, and represents an observation relative to a particular situation in the world [Rig98]. In this setting, each observation is considered to be independent each other, therefore assuming that attribute values of an observation are not affected by attributes of other observations. This is clearly in contrast with principles regulating geographical environments stating that some form of spatial correlation holds for spatial object. In particular, a commonly agreed rule is the *first Law of Geography*, stating that "everything is related to everything else, but near things are more related than distant things" [Tob79].

This discussion highlights that different approaches exist in literature, each of them clustering objects looking at data in different perspectives and, hence, focussing on some aspects of the problem and neglecting other ones considered to be secondary.

To overcome such limitations, we present a multi-relational clustering method, named CORSO (Clustering Of Related Structured Objects), that is also able to take into account relations existing between objects in cluster detection. In particular, it follows a spatial clustering approach while detecting groups of objects in order to take into account relations between them. Following this principle, two objects can belong to the same cluster only if a (indirect) relational path exists between them. CORSO resorts to graph-based partitioning methods in order to have an easy and powerful way of representing data and relations in a structured fashion. In this perspective, our basic goal can be considered as a graph partitioning problem, where links do not have a predominant role but importance is given to structural similarities as well. Finally, it takes from MRDM the ability of "looking inside the objects" by evaluating their homogeneity according to their structural similarity. This means that objects are considered to be composed by sub-elements, generally heterogeneous, related each others. To deal with structured objects, CORSO resorts on an ILP inductive learning system, named *ATRE* (Apprendimento di Teorie Ricorsive da Esempi) [Mal03], that is used to induce models over cluster candidates in order to evaluate clusters homogeneity [MAVL05].

1.2 Structure of the thesis

This dissertation is organized as follows. Chapter 2 presents an overview of clustering methods working on spatial data. In particular, the spatial data mining environment is presented, focussing on prominent aspects of data we have to deal with and posing particular attention to the related issues to be faced. Further, the most representative methods in literature will be described highlighting, for each of them, the main peculiarities and the most important limits.

Similarly, in Chapter 3 basic concepts at the basis of Multi-Relational Data Mining are presented. Techniques belonging to this discipline resort to *Inductive Logic Programming* (ILP) in order to easily navigate relational data structure, since it is able to describe both data and relations in the same formalism. These aspects are deeply investigated in the chapter, and an ILP system, namely ATRE, is presented that is employed to deal with multi-relational aspects in our framework. Further, an overview of conceptual clustering approach is presented, according to the contribution that it provides to our work.

Chapter 4 is devoted to present CORSO, the central argument of this thesis, describing the motivations it originate from, its underlying idea and working, the issues related to the proposed method and the solutions adopted to face them.

In Chapter 5, some applications of CORSO are presented. Some of them concerns artificial data (such as LAN dataset) deliberately build to test our method in particular cluster configurations, such as concentric or (partially) overlapped clus-

ters other than irregular shaped clusters. On such data, a lot of existing methods badly perform. Some other applications concerns real-world data (such as OFANTO and ARPA datasets), characterized by a large number of observations and very complex data descriptions. The adoption of these datasets is interesting because it stresses the real mining capacity on huge amount of data and the scalability aspects of the method in exam.

Finally, in Chapter 6 conclusions are drawn and direction for future work are proposed.

Chapter 2

Clustering spatial data

This chapter introduces and illustrates the problem of clustering spatial data focussing on main aspects that characterize this environment.

In particular, mining spatial data is a more challenging task than mining generic data gathered from relational and transactional databases, since they are characterized by a twofold nature: they are described in terms of both descriptive attributes and spatial attributes. The former category includes the attributes usually found in classical data mining approach, while the latter category includes attributes related to their shape or their position. This adds to spatial data mining an additional degree of complexity with respect to generic data mining.

In the following section we will present these additional difficulties in manipulating spatial data. In Section 2.2 some well-known approaches available in literature to cluster spatial data will be discussed, presenting for each of them the most representative methods and illustrating their main strengths and drawbacks. In Section 2.3 we will introduce the graph-based learning approach, an alternative approach to mine patterns from data relying on graph-based structure imposed over data by their spatial relations. Finally, in Section 2.4 concluding remarks on this topic will be drawn.

2.1 Mining spatial data

Prominent advancement in data collection technology caused by the wide spread of bar code readers, satellites, magnetic and optical sensors, etc., has made data collection a common everyday routine. Thus, data collected has grown exponentially making it impossible for humans to intelligently and profitably analyse them. In addition, in last decades the advent of sophisticated instruments for remote sensing employed in different disciplines such as meteorology, archaeology and anti-terrorist investigation, to cite some examples, has caused a general need for being able to deal with spatial data.

Spatial data differ from generic data in that, in addition of descriptive attributes, they also have spatial information, that is information about their shapes, their sizes or their positions in a reference frame. In other words, spatial data also describes

information related to the space occupied by objects. Examples of spatial data are:

- regions in a nation, where each region is spatially described by its shape, latitude and longitude other than number of inhabitants, economy type, growth rate, and so on
- buildings in a city, where each building can be described in terms of shape of outside walls and position in the city (spatial attributes) other than number of floors, age and number of households (descriptive attributes)
- objects in an image, where each object is represented by a colour, a name, a category, but also by a 2D shape outlining it and a position in the pixel grid

According to how spatial information is present on the space, spatial data can be either discrete or continuous. *Discrete data* typically corresponds to single points in multi-dimensional space. It differs from non-spatial data in that it has a distance attribute that is used to locate the data in space. *Continuous data* spans over regions of space and might consist of map regions, medical images, star fields, etc. [Kol01]

Another useful taxonomy for spatial data was suggested by [Cre91] that distinguishes between three classes of spatial data, namely *lattice data* (that is discrete variation over space, with observations associated with regular or irregular areal units), *geostatistical data* (observations associated with a continuous variation over space, typically in function of distance) and *point patterns* (occurrences of events at locations in space).

Considering the way in which data are represented, we can have raster representation and vector representation.

Raster representation describes phenomena by dividing the world into regular areas (typically square cells) each of them having only one value (calculated as minimum, maximum, average or sum value over the entire area or as value in central point) for each attribute taken into account; this means that no spatial variation of attributes is present inside each area. In this form of representation, objects correspond to collections of areas and are represented by the points contained in them (see Fig. 2.1).

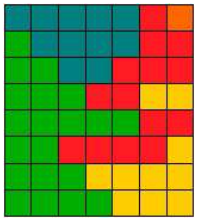


Figure 2.1: An example of raster representation

In *vector representation*, spatial objects consist of ordered sets of xy-coordinates that are used to draw points, lines, polylines, polygons and polyhedrons (see Fig.

2.2). Points belonging to these geometrical figures are used to represent the spatial object. Due to their vectorial nature, they can be easily scaled, translated or rotated, even if spatial relations between objects (e.g., overlap) are not explicitly expressed.

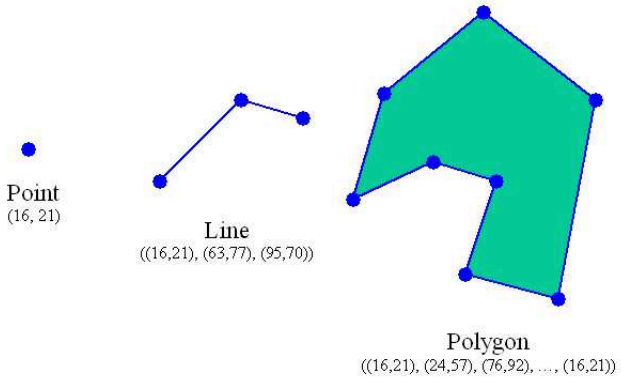


FIGURE 2.2: Some examples of vector representation

Raster representation, in comparison with vector representation, is composed by a simpler data structure since each spatial object is decomposed into a set of elementary regular areas but, on the other hand, this also implies a more imprecise geometry associated with each stored object. On the contrary, vector data can be easily and efficiently manipulated (scaled, rotated, translated) and can be efficiently stored without requiring high space to be allocated, with respect to raster data.

From the above discussion it is clear that classical DM approaches result to be inadequate to deal with spatial data since they are able to manipulate only the descriptive part of available information. What we need is an enhanced version of DM that must also take into account spatial attributes in the search for patterns.

This has given rise to *spatial data mining*, that can be defined as the extension of data mining from relational and transactional databases to *spatial databases*, that is, databases storing and being able to manipulate both descriptive and spatial attributes.

Spatial data mining differs from regular data mining in parallel with the differences between spatial and non-spatial data. The attributes of a spatial object stored in a database may be affected by the attributes of the spatial neighbors of that object. In addition, patterns relating spatial attributes to descriptive attributes may be exactly the information that we need to know [Fay96].

Spatial data mining methods can be applied to extract interesting and regular knowledge from large spatial databases. In particular, it can be used for understanding spatial data, discovering relationships between spatial and non-spatial attributes, understanding data organization in spatial databases, capturing the general characteristics in fairly simple and concise manner, etc. [Adh96]. This methods not only can profitably applied in Geographic Information Systems (GIS), but also in every other human discipline such as bio-informatics, medical analisys, robot

navigation and document image understanding. For instance, document image understanding could exploit the powerful capabilities of spatial data mining to organize and mine important patterns from digitalized documents. To this aim, each document page could be considered as a physical space containing spatial objects representing paragraphs, figures and tables arranged in the page. Spatial data mining could be able to highlight useful patterns hidden in the documents.

The presence of a spatial dimension in the data adds substantial complexity to the data mining tasks. In the following subsections we will present the three main issues to be faced when dealing with spatial data.

2.1.1 Implicit spatial relations

Spatial objects are characterized by a geometrical representation and position with respect to a reference system. The former implicitly defines a number of spatial attributes (e.g., orientation or area), while the latter generates a number of implicit spatial relations [CA06] belonging to one of the following three basic types [EKS01]:

- distance relations
- direction relations
- topological relations

Distance relations are relations comparing the distance of two objects with a given constant using one of the arithmetic operators.

Definition 2.1 *Let dist be a distance function, let σ be one of the arithmetic predicates $<$, $>$, $\leq$, $\geq$ or $=$, let c be a real number and let A and B be spatial objects. Then a* ***distance relation*** *$distance_{\sigma c}(A, B)$ holds iff $dist(A, B)\sigma c$.*

For instance, the relation $distance_{<50}(X, Y)$ holds for each pair of spatial objects $< X, Y >$ that are distant each other less than 50 units.

Direction relations are relations expressing information about the relative position of an object with respect to another one. Since these relations are not symmetric, it is important to distinguish between the *source object* O_1 and the *destination object* O_2. There are several ways to define direction relations depending on the number of considered points per object. By considering one representative point per object, some direction relations can be defined as follows:

Definition 2.2 *Let $rep(A)$ be a representative point in the source object A. Then:*

- *$northeast(B, A)$ holds iff $\forall b \in B : b_x \geq rep(A)_x \wedge b_y \geq rep(A)_y$*
  southeast, southwest and northwest are defined analogously
- *$north(B, A)$ holds iff $\forall b \in B : b_y \geq rep(A)_y$*
  south, west and east are defined analogously

where for each point $p = (p_x, p_y)$, p_x and p_y denote the coordinates of p

For instance, according to the above definition and considering European nations as spatial objects, we can state that the following direction relations hold: $northwest(France, Italy)$ and $east(Germany, Spain)$.

The third type of spatial relations is represented by *topological relations*. These are relations that are preserved if both objects are rotated, scaled or translated simultaneously (that is, they are invariant under topological transformations) and their definitions are based on the concepts of *boundaries*, *interiors* and *exteriors* of the two related objects.

Definition 2.3 *The* ***topological relations*** *between two objects A and B are derived from the nine intersections of the interiors, the boundaries and the exteriors of A and B with each other. The relations are: $disjoint(A, B)$, $meets(A, B)$, $overlaps(A, B)$, $equals(A, B)$, $covers(A, B)$, $covered\_by(A, B)$, $contains(A, B)$ and $inside(A, B)$.*

In particular, the above criterion employed to formally describe topological interactions between objects relies on the *9-intersection model* proposed by Egenhofer [Ege91]. Starting from the partitioning of the points in the space in three groups with respect to a fixed object (*interiors*, *exteriors* and *boundaries* of the object), this model defines a 3×3 matrix (*9-intersection matrix*) representing the relation between two spatial objects. In such matrix, a cell contains 1 iff the intersection of corresponding point sets is not-empty, 0 otherwise.

For instance, the 9-intersection matrices corresponding to three of the topological relations depicted in Fig. 2.3 are reported in Fig. 2.4.

Relations between A and B	A = B =
disjoint(A, B)	
meets(A, B)	
overlaps(A, B)	
equals(A, B)	
covers(A, B)	
covered_by(A, B)	
contains(A, B)	
inside(A, B)	

FIGURE 2.3: Topological relations between two spatial objects

However, this model often results to be unsatisfactory in many applications, since the end-user of a DM system is interested in the discovery of human-interpretable properties and relations between spatial objects rather than relations between their geometrical representations. For instance, a geographer could be interested to know that an urban area is crossed by a river without caring about the real geometrical shapes used to represent them.

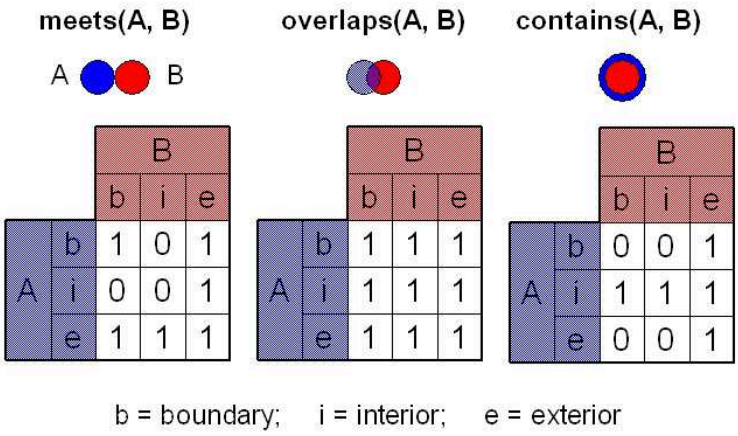


Figure 2.4: Some examples of 9-intersection matrices

Topological, distance and direction relations may be combined by the logical operators $\wedge$ and $\vee$ to express a *complex spatial relation*.

Definition 2.4 *If r_1 and r_2 are spatial relations, then $r_1 \wedge r_2$ and $r_1 \vee r_2$ are also spatial relations, and are called* ***complex spatial relations****.*

For instance, according to the above definition, a complex spatial relation is:

$$distance_{\leq 100}(X, Y) \wedge (northeast(X, Y) \vee southwest(X, Y))$$

Anyway, the existence of spatial relations would be useless if we are not able to perform spatial queries in order to retrieve objects on the basis of some spatial property; for instance, we could need to know "what are the streets of Rome *meeting* the Colosseum". Since spatial relations are not explicitly stored in the database, their extraction in a preprocessing step could be unfeasible because the potential volume of data to elaborate may be very large. Consequently, the spatial relations have to be computed on the fly. To facilitate this computation, spatial data must be stored in an appropriate manner and particular types of indexes appositely made to work with spatial attributes, called *spatial indexes*, are needed to be implemented in spatial DBMS. Examples of well-known spatial index are R-tree, R^+-tree and quadtree [Sam95].

In order to mitigate the efficiency problems in the computation on the fly of relations among spatial objects, some methods (typically belonging to spatial clustering algorithms family) perform a single scan of the database grouping the objects on the basis of local conditions. Basically, these methods, called *single-scan clustering algorithms*, retrieve the neighborhoods of the objects in the database and work on these pieces of data at a time [EFKS00]. Given a spatial object o, its *neighborhood* is calculated by retrieving all the objects in relation with o according to some spatial relation. Obviously, single-scan clustering algorithms are very efficient if the retrieval of the neighborhood of an object is efficiently supported by the DBMS, and then, if the spatial indexes implemented by the DBMS perform good. If the

average runtime complexity of a region query is $O(\log n)$, then for a database of n objects the overall runtime complexity of a single-scan algorithm is $O(n \log n)$ [EKSX98].

Neighborhoods are calculated by resorting to concepts of neighborhood graphs and neighborhood paths. In the following, we formally define these concepts.
Let *neighbor* be a spatial relation and DB be a database of objects:

Definition 2.5 *A **neighborhood graph** $G^{DB}_{neighbor} = (N, E)$ is a graph where*

- *each node $n \in N$ corresponds to a spatial object $o \in DB$*
- *each edge $e \in E \subseteq N \times N$ connects two nodes $n_1, n_2 \in N$ iff $neighbor(n_1, n_2)$ holds.*

If we denote with $\#N$ the cardinality of N and with $\#E$ the cardinality of E, then $F = \#E/\#N$ denotes the average number of edges of a node. F is called "fan out" of the graph.

Neighborhood graphs, also called *proximity graphs* [Tou91], because they capture proximity between points by connecting nearby points with a graph edge.

Since the number of nodes composing a neighborhood graph corresponds to the number of spatial objects in the database and each node is associated with only one spatial object, in this section we will use the symbol N to denote the set of nodes in the graph.

Definition 2.6 *A **neighborhood path** is a sequence of nodes $[n_1, n_2, \ldots n_k]$, where $neighbor(n_i, n_{i+1})$ holds $\forall n_i \in N, 1 \leq i < k$. The number k of nodes is called **length** of the neighborhood path.*

Definition 2.7 *A **neighborhood path** $[n_1, n_2, \ldots n_k]$ is **valid** iff $\forall i \leq k, j < k$: $i \neq j \Leftrightarrow n_i \neq n_j$.*

Valid neighborhood paths are paths without cycles. In spatial data mining, we are only interested in valid neighborhood paths, because paths containing cycles and visiting the same node more than once will not produce useful patterns [EFKS00].

Lemma 2.1 *The expected number of neighborhood paths of length k starting from a given node is f^{k-1} and the expected number of all neighborhood paths of length k is then $n * f^{k-1}$.*

Both neighborhood graphs and neighborhood paths are used to reduce the computational complexity of the extraction of spatial relations among data. Some neighborhood graphs well-known in literature are [And01]:

- *Delaunay triangulation*: $DT(N)$
- *Nearest neighbor graph*:
  $NNG(N) = \{Edge(n_1, n_2) | n_1, n_2 \in N \wedge B(n_1, \delta(n_1, n_2)) \cap N = \oslash\}$
- *Minimum spanning tree*: $MST(N)$

- *Relative neighborhood graph*:
  $RNG(N) = \{Edge(n_1, n_2) | n_1, n_2 \in N \wedge L_2(n_1, n_2) \cap N = \oslash\}$

where: N denotes a set of nodes corresponding to spatial objects; $Edge(n_1, n_2)$ holds iff the nodes n_1 and n_2 have a common edge; $\delta(n_1, n_2)$ is the distance between n_1 and n_2 according to a given metric; $B(n_1, r) = \{n | \delta(n_1, n) < r\}$ and $L_2(n_1, n_2) = B(n_1, \delta(n_1, n_2)) \cap B(n_2, \delta(n_1, n_2))$.

The presented neighborhood graphs are related each other in that they belong to the same hierarchy:

$$NNG \subseteq MST \subseteq RNG \subseteq DT$$

Another way to formally represent neighborhoods is the *spatial weights matrix* (W). This is a square matrix of dimension equal to the number of observations (n), in which each row and matching column correspond to an observation pair i, j. The generic entry $w_{i,j}$ takes a non-zero value (1 for a binary matrix, or any another positive value for general weights) when observations i and j are considered to be neighbors, and a zero value otherwise.

2.1.2 Spatial autocorrelation

Another challenge to face when dealing with spatial data is represented by *spatial autocorrelation*. This effect, resulting directly from Tobler's First Law of Geography, stating that "*everything is related to everything else, but near things are more related than distant things*" [Tob79], states that similar values for a variable will tend to occur in nearby locations.

For example, a high crime neighborhood in a city will often be surrounded by other high crime areas, or a low income county in a remote region is easy to be adjacent to other low income counties. This means that it may happen that the cause of an effect observed in a given object is located in one of its neighbors, and vice versa. For instance, a high rate of cancer diseases in a region can be caused by an oil refinery placed in a nearby area.

As a consequence, in contrast to mining in relational databases, spatial DM algorithms have to consider the neighbors of objects in order to extract useful knowledge. This is necessary because the attributes of the neighbors of some object of interest may have a significant influence on the object itself. This leads to distinguish the spatial objects in two categories:

- *reference objects*, that is the objects we are directly interested in our analysis
- *task-relevant objects*, that is the objects that are not directly concerned in our analysis, but that must be taken into account because can affect the properties of the reference objects

Spatial autocorrelation can be specialized in two alternative phenomena:

- *positive spatial autocorrelation*, when large (small) values are systematically surrounded by other large (small) values

- *negative spatial autocorrelation*, when large values are surrounded by small values and vice versa (leading to a checkerboard pattern of values)

2.1.3 Multiple levels of granularity

The third issue related to mining spatial data is that spatial objects can often be organized in hierarchies of classes. By navigating such hierarchies, it is possible to view the same spatial object at different levels of details (or granularity), each of them allowing to discover particular patterns that could be difficult to mine at a different level. For instance, looking at a city as a whole one could discover that cities with few public parks have, in general, a high number of citizens suffering from diabetes (because they dedicate little or no time to physical activities and are obese); by considering cities as composed by quarters, one could discover that historical centres are populated by inhabitants characterized by both a low income rate and a low cultural level, thus discovering that low income rate is directly related to the cultural level of people.

It is clear that data mining methods should be able to explore the search space at different granularity levels in order to facilitate the discovery of useful patterns. In case of granularity levels defined by a containment relation, this corresponds to exploring both global and local features of the same observation.

Very few data mining techniques automatically support this multiple-level analysis. In general, the user is forced to repeat independent experiments on different representations, but in this way results obtained for each granularity level cannot be used to enrich the information available at different levels. In addition, spatial autocorrelation relationships change among different spatial granularity levels [OT79, Arb89]. Therefore, the discovery of patterns at different levels of granularity is another challenge for research in spatial data mining.

2.2 Spatial clustering methods

Spatial clustering can be defined as the process of grouping a set of spatial objects into classes (or *clusters*) so that objects within a cluster have high similarity in comparison to one another, and are dissimilar to objects in other clusters. Differently from generic clustering, spatial clustering evaluates the similarity according to spatial features of data and, hence, usually "similarity between two objects" is translated with "spatial proximity of two objects". Proximity definition varies according to the relation considered in the particular application. Therefore, if the considered relation is "Euclidean distance", then the nearby objects are grouped in the same cluster, while if the relation taken into account is "adjacency", then objects are grouped according the neighbors they are in touch.

In this section some well-known approaches available in literature to cluster spatial data will be discussed, presenting for each of them the most representative methods and illustrating their main strengths and drawbacks.

It is important to remark that the perfect method that is suitable for each situation, in each domain and for each type of data does not exist. However, in

order to choose the best suited algorithm for a particular application, the following factors have to be considered [HKT01]:

- *application goal*: the goal of the application will often affect the type of clustering algorithm being used. For instance, if we are asked to group customers of a supermarket chain in order to detect the best locations for setting up the stores, the better methods to use are the partitioning ones (such as *k-means* and *k-medoids*) since we are interested in having the distance between each customer and the store (centre of the cluster) as short as possible. If we want to discover the arrangement of a particular animal species, the partitioning-based methods are not the best choice, but it would be better to use the density-based approach.

- *trade-off between quality and speed*: as a rule of thumb, there will usually be trade-offs between the speed of a clustering algorithm and the quality of the produced results. A clustering algorithm that produces good quality clusters may be unable to handle large datasets, thus being suitable only for applications with relative few objects. To handle large datasets, a common approach is to perform some form of compression on data before clustering, but, since compression is usually lossy, the quality of resulting clusters will often drop.

- *characteristics of the data*: the characteristics of data are often decisive in the choice of the algorithm to adopt. These characteristics include:

  - *the types of data attributes*: the similarity between two spatial objects is evaluated by comparing their data attributes (numeric, binary, categorical, ordinal). It is obvious that clustering methods unable to deal with one of these attributes types prevent it from being used in applications where that type of attribute is important. For instance, most of spatial clustering methods are perfectly able to deal with numeric attributes but flounders with the other types.
  - *dimensionality*: many algorithms which perform well on low-dimensional data (that is, data objects with a low number of attributes), degenerate when the number of attributes increases. This deterioration can be revealed by either an increase in running time or a decrease in cluster quality, or both.
  - *amount of noise in data*: some clustering algorithms are very sensitive to noise and outliers (e.g., *k-means* and *k-medoids*), while other algorithms have been deliberately designed to be noise insensitive.

2.2.1 Partitioning methods

Partitioning approach characterized early studies in clustering and are still one of the most cited approach in literature.

Given a set N of spatial objects defined in a D-dimensional space (that is, described by D attributes), an input positive integer k and a *similarity function*

saying how much two spatial objects are similar each other, partitioning algorithms organize the objects into k clusters such that the similarity function of each object with respect to its cluster representative is minimized. As a cluster representative can be used the cluster centre, the most centrally located object in the cluster or a probability distribution representing the cluster.

Usually, similarity function corresponds to Euclidean distance, so "minimize the similarity function between a given object and its cluster representative" means to associate the object to the cluster having the closest representative.

Methods belonging to this category are affected by two common drawbacks that discourage their employment in real applications:

- objects must be arranged into a fixed number of clusters that must be specified a priori by the user
- they are able to discover only convex shapes, since they associate objects to the most similar (or closest) cluster representative

These limitations are very restrictive because social and environmental phenomena are seldom characterized by spherical shapes. In addition, the number of groups of similar objects is itself a nugget of useful knowledge hidden in data that user want to know.

Furthermore, the algorithms belonging to this approach adopt an *iterative relocation technique* in the clusters detection procedure. In particular, as first step, they choose k objects as the initial solution, that is, as the representatives of the k clusters. Then, they assign each of the remaining objects to the most similar cluster representative. After all the objects have been assigned to a cluster, they recompute the cluster representative according the memberships of the objects. If any change occurs in the cluster representative assignment, than the membership of the objects is recalculated and the process restarts; otherwise, the detected clusters represent the final results and the procedure terminates. This procedure is reported in Algorithm 2.1.

Algorithm 2.1 Generalized iterative relocation approach

```
1: function IterativeRelocation(k,DB)
2:     solution ← arbitrarily choose k objects in DB as the initial clusters representatives;
3:     repeat
4:         (re)assign each object in DB to the most similar object in solution;
5:         solution ← update the clusters representatives according to the current objects membership;
6:     until no change occurs in solution;
7:     return solution;
8: end function
```

A lot of algorithms have been developed that belongs to partitioning approach, but almost all of them can be considered variations of two basic and very simple algorithms: *k-means* and *k-medoids*.

The *k-means* algorithm [Mac67] uses the centre of the cluster as cluster representative, that is calculated as the mean value of the objects currently assigned to the cluster. Typically, the objective criterion adopted to search for a good solution is to minimize the squared-error function, defined as follows:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - m_i|^2$$

where: x represent a generic object of DB assigned to the cluster C_i, and m_i is the centre of the cluster C_i.

It results to be relatively scalable and efficient, because its computational complexity is $O(nkt)$, where n is the number of objects ($n = \#N$), k is the number of the clusters, and t is the number of iterations (usually $t \ll n$).

An alternative version of k-means is the *k-modes* method [HN03] that, differently from k-means, deals with categorical attributes, by using a special definition of the similarity function comparing this kind of attributes and a frequency-based method to update the cluster representatives (called *modes*). K-means and k-modes methods have been successively integrated to be able to deal with both numerical and categorical attributes at the same time, resulting in a new method called *k-prototypes* method.

However, k-means algorithm is affected by two additional limitations:

- it is very sensitive to noise and outliers: a small number of them can substantially change the cluster centres
- cluster representatives are often scarcely meaningful, since the cluster centres rarely correspond to objects in the database.

In order to mitigate such drawbacks another method, called *k-medoids*, has been proposed in [KR90] that uses as a cluster representative the most centrally located object (*medoid*) in a cluster instead of its centre. In particular, taking into account the algorithm 2.1, at each iteration only one medoid is replaced by one of the non-medoid objects. As a consequence, to find the best solution k-medoids performs much more iterations in comparison with k-means because only one cluster representative is changed at each step.

To reduce the number of iterations, a variant of k-medoids, namely *PAM* (*Partitioning Around Medoids*), was proposed that for each iteration tries to replace each medoid with one of the other $(n-k)$ objects in the database. The set of best objects for each cluster in one iteration forms the medoids for the next iteration. Since in each iteration PAM tries $k(n-k)$ substitutions, because it tries to replace each medoid with a non-medoid, and for each substitution it performs $(n-k)$ operations to evaluate the squared-error function E, the computational complexity of PAM in one iteration results to be $O(k(n-k)^2)$.

It is clear that PAM is computationally costly, so its application to large datasets is prohibitive. In order to be able to deal with large amount of data, *CLARA* (*Clustering LARge Applications*) was developed as an improvement of PAM. The basic idea is to choose cluster representatives from a small portion of sampled data, instead of choosing them from the entire dataset. In this way, medoids are calculated

as approximation of the medoids calculated by PAM, even if the squared-error function is calculated by considering all the objects in the dataset. Anyway, CLARA cannot find the best clustering if any best k medoids (that would be selected by PAM) is not among the sample chosen by CLARA. This precision loss is traded off with a more efficient computation, since the complexity of each iteration becomes $O(ks^2 + k(n-k))$, where s is the size of the sample.

As stated above, the main drawback of CLARA is that the solution found by this method is not the best solution if any sampled medoid is not among the best k medoids. This happens because once a data sample is chosen for the medoids selection, no swap is performed between sample set and remaining objects. *CLARANS* (*Clustering Large Applications based upon RANdomized Search*) was proposed by Ng and Han [NH94] that, when searching for a better centre, tries to find a better solution by randomly picking one of the k centres and trying to replace it with another randomly chosen object from the other $(n-k)$ objects.

Differently from above mentioned methods, the *EM* (*Expectation Maximization*) algorithm [BFR98] represents each cluster using a probability distribution instead of a cluster representative. In particular, the Gaussian probability distribution is used because any density distribution can be effectively approximated by a mixture of Gaussian distributions. Furthermore, another difference of EM with respect to the other partitioning methods is that it assume that each object is member of each cluster with a certain probability of membership. As a objective criterion, EM clustering tries to maximize the log likelihood of the mixture model computed as:

$$E = \sum_{x \in DB} \log(P(x))$$

where $P(x)$ represents the combination of the effect on the same object x of the different cluster distributions (mixture model probability density function).

2.2.2 Hierarchical methods

Differently from the other clustering approaches, hierarchical methods create a *dendrogram* of spatial objects, that is a tree structure where each non-leaf node is composed by the same elements composing its children nodes. In this way, the result of a clustering task is not a partition of the dataset, but a hierarchy of clusters where each level describes a partition of source data. The dendrogram can be built according to two approaches:

- *Agglomerative hierarchical clustering*: a bottom-up approach is performed in the dendrogram construction. Starting by placing each object in its own cluster, at each iteration the two most similar clusters (according to the considered similarity function) are merged together into a new cluster until all of the objects are in a single cluster or until a termination condition holds. A method belonging to agglomerative approach is *AGNES* (*AGlomerative NESting*).

- *Divisive hierarchical clustering*: the dendrogram is built according to a top-down strategy. It starts by considering each object belonging to the same cluster and, at each iteration, one of the available clusters is split into smaller

clusters according to some measure, until each object is in one cluster or a termination condition holds. A method belonging to divisive approach is *DIANA* (Divisive ANAlysis).

A schematic representation of the dendrogram construction in both the approaches is reported in Fig. 2.5.

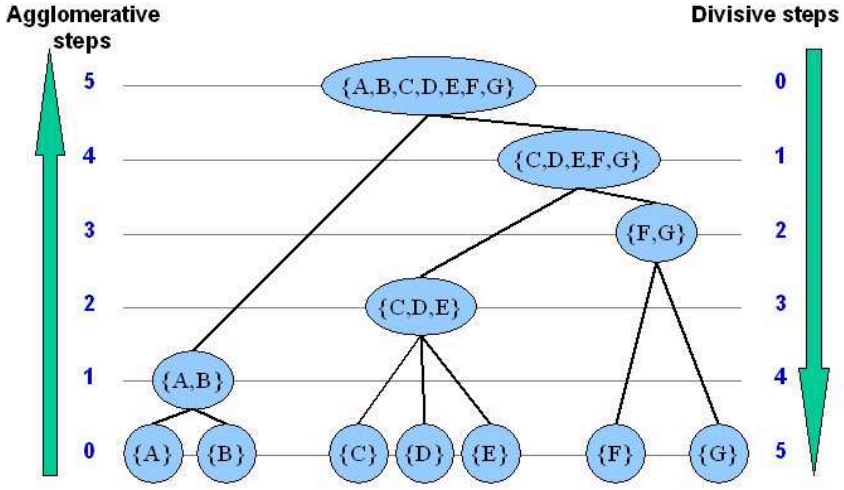


Figure 2.5: Hierarchical clustering: agglomerative and divisive approaches

Termination criteria adopted to decide when to stop the generation of dendrogram are generally based on the number of detected cluster or the distance between clusters. In the latter case, four widely used measures are:

- *Minimum distance*: $d_{min}(C_i, C_j) = min_{p \in C_i, q \in C_j} |p - q|$
- *Maximum distance*: $d_{max}(C_i, C_j) = max_{p \in C_i, q \in C_j} |p - q|$
- *Mean distance*: $d_{mean}(C_i, C_j) = |m_i - m_j|$
- *Average distance*: $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{q \in C_j} |p - q|$

where: $|p - q|$ is the distance between the objects p and q, m_i is the mean for the cluster C_i and n_i is the number of objects in C_i.

Both agglomerative and divisive approaches suffer from their inability to perform adjustment once a merge or a split decision is performed. This means that, the algorithm can neither undo the operations performed in previous steps nor perform object-swapping between clusters. Thus, bad decisions taken at some step (that is, a bad split of a cluster in case of divisive approach or a bad merge of two clusters in the agglomerative approach) lead to low-quality clusters. Furthermore, hierarchical algorithms that rely on distance measures when deciding if merge or split clusters usually perform well only on clusters with spherical shapes. In addition, generally the hierarchical approach does not scale well because in order to evaluate if a merging or a split has to be performed a large number of objects must be compared.

In order to be able to deal with large databases, an alternative hierarchical method has been proposed in [ZRL96], namely *BIRCH* (*Balanced Iterative Reducing and Clustering using Hierarchies*). The idea is to arrange the objects into small subclusters, and perform clustering with these subclusters. Since the number of subclusters is much lesser than the number of available objects, the clustering task can be performed efficiently because all information can be stored in main memory. Each subcluster is represented by a Clustering Feature (CF), that is a triplet summarizing the information about the objects belonging to the subcluster. Each CF is defined as $CF = (N, \vec{LS}, SS)$, where N is the number of objects in the subcluster, $\vec{LS}$ is the linear sum on N objects, and SS is the square sum of objects.

These CF are stored in a tree, called *clustering feature tree* (CF-tree), in order to apply some clustering technique to them. In such hierarchy, the non-leaf nodes are calculated as the sum of the CFs of their children. The size of the nodes composing the tree can be altered by means of two parameters: a *branching factor* B, that specifies the maximum number of children per non-leaf node, and a *threshold* T, defining the maximum diameter of subclusters stored at the leaf nodes.

The CF-tree is build dynamically as objects are read from the database. In particular, as an object is read, it is inserted to the closest subcluster (corresponding to a leaf node). If the diameter of subcluster exceed T, than the corresponding leaf node is split, propagating the split to the upper levels of hierarchy in case of the node split makes its parent diameter exceeding T. Anyway, after an object insertion in a leaf node, its CF and the CFs of all the nodes belonging to the path from the leaf node to the root are update, accordingly. After the CF-tree has been built, a clustering algorithm is applied to leaf nodes.

A drawback of this method is that, if the real clusters have not spherical shapes, it does not perform well since it controls the boundaries of the CF-tree nodes (that represent the element actually clustered) by means of their diameter. On the other hand, BIRCH results to be very efficient (its computational complexity is $O(n)$), since it read the objects from the database only to build the CF-tree and every other operation is performed by resorting to this structure. Furthermore, BIRCH is an incremental method, therefore it can easily and dynamically fit to data as they are read.

Another well-known method belonging to hierarchical clustering is *CURE* (*Clustering Using REpresentatives*) [GRS98]. It proposes solutions to three important issues generally affecting hierarchical clustering algorithms, namely detection of non-spherical clusters, robustness with respect to outliers, and being able to deal with large databases.

Firstly, instead of representing a cluster with a single element, being it the cluster centre or another object in the cluster, CURE associate a cluster with a fixed number of representative objects. These representatives are calculated by selecting well-scattered objects for the cluster and shrinking them toward the centre by a specified fraction (*shrinking factor*). At each step of the algorithm, the two clusters with the closest pair of representatives are merged. This allows CURE to find cluster shapes that are more suitable for non-spherical clusters.

Secondly, the shrinking step of the cluster representatives makes CURE more robust to outliers, since in this way outliers are cut out from the detected clusters.

Finally, in order to be able to manipulate a large number of objects, CURE combines a random sampling step with a partitioning step: the algorithm draws a random sample of data, fragments it in a set of partitions and then partially clusters them. The result of such partial clustering is further clustered to yield the final clusters.

Even if CURE tries to face with the issues discussed above, some problems remain unsolved. First, clustering results are heavily affected by the parameter setting, namely the sample size, the desired number of clusters and the shrinking factor. Second, CURE does not handle categorical attributes.

The latter limitation is overcome by *ROCK* [GRS00], that employs the concept of links and not distances when merging clusters. More specifically, it measures the similarity of two clusters by computing the interconnectivity of two clusters, defined as the number of objects from different clusters that share some neighbors.

Hierarchical clustering methods examined till now only rely on either clusters proximity or clusters interconnectivity, but not on both these features.

CHAMELEON [KHK99] starts from this observation to define its way of work. In its clustering process, two clusters are merged if the interconnectivity and closeness (or proximity) between the clusters is comparable to the internal interconnectivity and closeness of objects within the clusters. Chameleon first uses a graph partitioning algorithm to cluster objects into a number of relatively small subclusters. Then, it applies an agglomerative hierarchical approach to these subclusters in order to detect final clusters. In the agglomerative step, two clusters are merged if their relative interconnectivity and relative closeness are high, that is, the algorithm selects to merge clusters that are well inter-connected and close with respect to the internal interconnectivity and closeness of the clusters. In this way, this algorithm can easily adapt to the internal characteristics of the clusters being merged and is more facilitate in the discovery of natural and homogeneous clusters. Experimental results presented by the authors show that Chameleon is able to discover more natural cluster than DBSCAN (see Section 2.2.3), since it is able to dynamically fit internal cluster characteristics. However, the processing cost for high-dimensional data may require $O(n^2)$ time for n objects in the worst case.

2.2.3 Density-based methods

Most of previously presented methods, being based on evaluation of the distance between objects, performs well in the detection of spherical-shaped clusters while encounter difficulties in the discovery of clusters with arbitrary shapes. In order to overcome such limitation, a new family of clustering methods has been developed that is based on the concept of *density*. Here, clusters are regarded as dense regions, that is, regions characterized by an high number of spatial objects. These dense regions are separated each others by regions of low density and containing noise. By relying on the notion of spatial density instead of distance, density-based methods

have all the necessary requirements to detect clusters of arbitrary shapes filtering out noise.

The most known method belonging to density-based clustering is *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*) [EKSX96]. It requires two input parameters, ϵ and $MinPts$, and detects and grows regions that are sufficiently dense. These parameters are used to determine the density of regions in the space by means of two basic concepts: $\epsilon - neighborhoods$ and *core objects* (see Fig. 2.6).

The $\epsilon - neighborhood$ of an object $p \in DB$ is the neighborhood of p within the area delimited by the circle of radius ϵ.

A *core object* is an object in the database having at least $MinPts$ neighbors in its $\epsilon - neighborhood$.

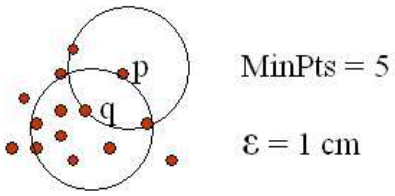


FIGURE 2.6: DBSCAN: the input parameters

Cluster detection in DBSCAN is formally defined as follows:

Definition 2.8 *The* ϵ***-neighborhood*** *of a point p, denoted by* $N_\epsilon(p)$*, is defined by* $N_\epsilon(p) = \{q \in DB | dist(p,q) \leq \epsilon\}$.

Definition 2.9 *A point p is* ***directly density-reachable*** *from a point q w.r.t.* ϵ *and* $MinPts$ *if:*

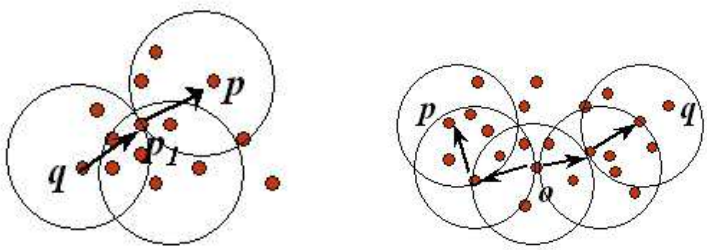
1. $p \in N_\epsilon(q)$ *and*
2. $|N_\epsilon(q)| \geq MinPts$ *(core point condition)*

Definition 2.10 *A point p is* ***density-reachable*** *from a point q w.r.t.* ϵ *and* $MinPts$ *if there exists a chain of points* $q = p_1, p_2, \ldots p_n = p$ *such that* $\forall i \in [1..n-1] : p_{i+1}$ *is directly density-reachable from* p_i *w.r.t.* ϵ *and* $MinPts$ *(see Fig. 2.7.a).*

Definition 2.11 *A point p is* ***density-connected*** *to a point q w.r.t.* ϵ *and* $MinPts$ *if there exists a point o such that both p and q are density-reachable from o w.r.t.* ϵ *and* $MinPts$ *(see Fig. 2.7.b).*

Definition 2.12 *Let DB a database of points. A* ***cluster*** C *w.r.t.* ϵ *and* $MinPts$ *is a non-empty subset of DB satisfying the following conditions:*

1. $\forall p, q \in DB$*: if* $p \in C$ *and q is density-reachable from p w.r.t.* ϵ *and* $MinPts$*, then* $q \in C$ *(Maximality condition)*



(a) p and q are density reachable (b) p and q are density connected

Figure 2.7: The density-reachable and density-connected concepts

2. $\forall p, q \in C$: *p is density-connected to q w.r.t.* ϵ *and* $MinPts$ *(Connectivity condition)*

Consequently, the following lemma holds:

Lemma 2.2 *Let p a point in DB and* $|N_\epsilon(p)| \geq MinPts$*. Then the set* $C = \{o \in DB | o$ *is density-reachable from p w.r.t.* ϵ *and* $MinPts\}$ *is a cluster w.r.t.* ϵ *and* $MinPts$*.*

DBSCAN discovers clusters by checking the $\epsilon - neighborhood$ of each object in the database. If the $\epsilon - neighborhood$ of an object p contains a number of objects equal or greater than $MinPts$, then p is considered to be a core object and a new cluster containing p and its $\epsilon - neighbors$ is created; otherwise, p is labelled as *noise object*. After a cluster has been detected, an expansion step is performed in which the algorithm tries to expand the cluster just created. The expansion is performed by repeating the $\epsilon - neighborhood$ detection on each $\epsilon - neighbors$ of p. When no more core objects are found in this growing process, another object in the database is chosen for the detection of another cluster. During the expansion step of the cluster C, if a core object q is encountered that already belongs to another cluster D, the addition of q in C corresponds to a merge of C and D. The algorithm stops when no more unlabelled objects exist in the database. The clustering detection procedure discussed above is reported in Algorithm 2.2.

DBSCAN is really able to detect clusters of arbitrary shapes and results to be insensitive to noise since it isolates outliers in a special class enclosing the objects not assigned to any cluster, but an important limitation of this method is that it can only work with point objects, that is, objects without a shape. To overcome such limitation, a generalization of this algorithm was proposed by Sander et al., namely *GDBSCAN* (*Generalized Density Based Spatial Clustering of Applications with Noise*) [SEKX98], that can cluster point objects as well as spatially extended objects according to both their spatial and non-spatial attributes. In particular, the generalization concerns two important aspects of the clustering algorithm: any notion of neighborhood cardinality can be used other that the canonical one adopted by DBSCAN, and any definition of neighborhood can be used that is based on a binary predicate which is symmetric and reflexive.

Algorithm 2.2 The DBSCAN algorithm

```
1: function DBSCAN(ϵ, minPts, DB)
2:     ClustersSet ← ∅;
3:     for all p ∈ DB do
4:         Neighborhood ← DB.regionQuery(p, ϵ);
5:         if p.label = NONE then
6:             if Neighborhood.size ≥ minPts then
7:                 C.add(p);
8:                 p.label = C;
9:                 for all q ∈ Neighborhood do
10:                    C.add(q);
11:                    q.label = C;
12:                end for
13:                ExpandCluster(C, Neighborhood, ϵ, minPts, DB);
14:                ClustersSet ← ClustersSet ∪ {C};
15:            else
16:                NOISE.add(C);
17:                C.label ← NOISE;
18:            end if
19:        end if
20:    end for
21:    return ClustersSet;
22: end function

23: procedure ExpandCluster(C, Neighborhood, ϵ, minPts, DB)
24:    Seeds ← Neighborhood;
25:    while Seeds ≠ ∅ do
26:        q ← Seeds.getFirst();
27:        Seeds ← Seeds − q;
28:        QNeighbors ← DB.regionQuery(q, ϵ);
29:        if QNeighbors.size ≥ minPts then
30:            for all obj ∈ QNeighbors do
31:                if obj.label = NONE ∨ obj.label = NOISE then
32:                    if obj.label = NONE then
33:                        Seeds ← Seeds ∪ {obj};
34:                    end if
35:                    C.add(obj);
36:                    obj.label = C;
37:                end if
38:                C.add(q);
39:            end for
40:        end if
41:    end while
42: end procedure
```

Similarly to DBSCAN, the clustering mechanism adopted by GDBSCAN relies on the following definition:

Definition 2.13 *Let* NPred *be a binary predicate on DB which is reflexive and symmetric, that is,* $\forall p, q \in DB : NPred(p,p) \wedge (Npred(p,q) \Rightarrow Npred(q,p))$. *Then, the* ***NPred-neighborhood*** *of an object* $o \in DB$ *is defined as* $N_{NPred}(o) = \{o' \in DB | NPred(o,o')\}$.

It is easy to see how the $\epsilon - neighborhood$ adopted by DBSCAN is a particular case of the $NPred - neighborhood$, where $N_\epsilon(o) = \{o' \in DB | dist(o,o') \leq \epsilon\}$.

For instance, if the spatial objects correspond to polygons, possible $NPred$ predicates that can be used are *Intersects* and *SameShape*. Obviously, in this way we are not restricted to consider only spatial attributes, but we can also consider predicates involving non-spatial attributes or even combine different types of attributes. For instance, GDBSCAN could be applied to figures depicted in Fig. 2.8 where the $NPred$ predicate is "Overlaps", the $MinWeight$ predicate is "$Area(N) \geq 100$", and the detected cluster is outlined by the dashed line.

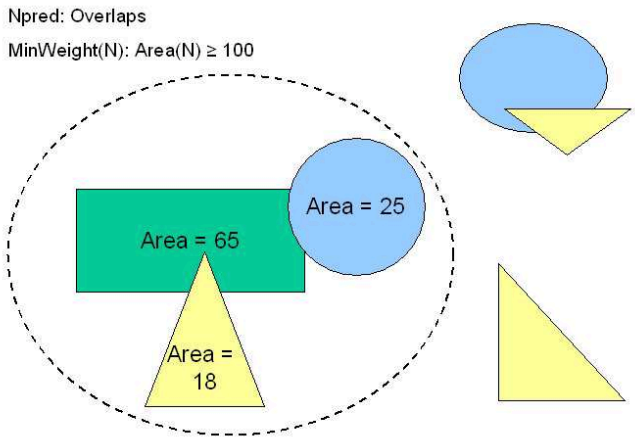


Figure 2.8: An application of GDBSCAN

The cardinality concept employed to define dense areas can be generalized as well.

Definition 2.14 *Let* $wCard$ *be a function from the subsets of* DB *into the non-negative real numbers,* $wCard : 2^{DB} \rightarrow \Re^+$, *and* $MinCard$ *be a positive real number. Then, the predicate* $MinWeight$ *for a set* S *of objects is defined to be true iff* $wCard(S) \geq MinCard$.

According to the above definition, $wCard(S) \geq MinCard$ is a generalization of the condition $|N_\epsilon(o)| \geq MinPts$, where *cardinality* is a special case of $wCard$.

In this way, other than considering the number of objects in the set, cardinality of a set of objects can be also defined in terms of values of non-spatial attributes, such as average value of incoming rates. Using non-spatial attributes as a weight for

objects one can define different densities, even if the objects are equally distributed in the space of the spatial attributes.

Consequently, the definitions 2.9, 2.10, 2.11, 2.12 can be generalized as follows:

Definition 2.15 *An object p is **directly density-reachable** from an object q w.r.t. $NPred$ and $MinWeight$ if:*

1. $p \in N_{NPred}(q)$ *and*
2. $(MinWeight(N_{NPred}(q)) = true$ *(core object condition)*

Definition 2.16 *An object p is **density-reachable** from an object q w.r.t. $NPred$ and $MinWeight$ if there exists a chain of objects $q = p_1, p_2, \ldots p_n = p$ such that $\forall i \in [1..n-1] : p_{i+1}$ is directly density-reachable from p_i w.r.t. $NPred$ and $MinWeight$.*

Definition 2.17 *An object p is **density-connected** to an object q w.r.t. $NPred$ and $MinWeight$ if there exists an object o such that both p and q are density-reachable from o w.r.t. $NPred$ and $MinWeight$.*

Definition 2.18 *Let DB a database of objects. A **cluster** (or **density-connected set**) C w.r.t. $NPred$ and $MinWeight$ is a non-empty subset of DB satisfying the following conditions:*

1. $\forall p, q \in DB$*: if $p \in C$ and q is density-reachable from p w.r.t. $NPred$ and $MinWeight$, then $q \in C$ (Maximality condition)*
2. $\forall p, q \in C$*: p is density-connected to q w.r.t. $NPred$ and $MinWeight$ (Connectivity condition)*

Similarly to DBSCAN, the following lemma holds:

Lemma 2.3 *Let p an object in DB and $MinWeight(N_{NPred}(p)) = true$. Then the set $C = \{o \in DB | o$ is density-reachable from p w.r.t. $NPred$ and $MinWeight\}$ is a density-connected set (or cluster) w.r.t. $NPred$ and $MinWeight$.*

Another extension of DBSCAN is represented by *DBRS* (*Density-Based spatial clustering method with Random Sampling*) [WH03] that, other than considering spatially nearby objects when calculating clusters, takes into account also non-spatial attributes. Thus, according to this approach, clusters are formed by objects that are close each others and that are considered to be homogeneous according to some non-spatial property. Such non-spatial homogeneity is calculated by a purity function. However, objects similarity is calculated by considering descriptive attributes that must be specified a-priori. This represent a substantial limitation, since user is asked to provide to the algorithm a precise indication of the attributes according to which objects are considered similar.

The quality of clusters obtained with DBSCAN (and its generalization, GDBSCAN) heavily depends on the user's ability to select good input parameters. This

often implies that a substantial number of trials need to be performed to perform parameters tuning.

To facilitate user in such task, a cluster ordering method called *OPTICS* (*Ordering Points to Identify the Clustering Structure*) was proposed by Ankerst et al. [ABKS99]. Even if OPTICS continues to require the input parameters ϵ and $MinPts$, it produces an ordering of the data points such that clustering results, for any lower value of ϵ and a similar value of $MinPts$, can be easily computed and visualized.

The idea stems from the observation that, given a $MinPts$ value, the number of detected clusters increases as the value of ϵ increases. In particular, if $\epsilon > \epsilon'$, then the set of clusters detected with ϵ' is completely contained in the set of clusters detected with ϵ. This happens for two reasons:

- core objects for ϵ might become non-core objects for ϵ' because they do not have at least $MinPts$ data objects in their $\epsilon' - neighborhood$
- non-core objects that are in the $\epsilon - neighborhood$ of some core objects for ϵ may become noise, either because they are not in the $\epsilon' - neighborhood$ of these core objects, or because these core objects have become non-core

Given an ϵ and $MinPts$ values, in order to produce a set of ordering of density-based clusters, for each object we need to store threshold values of *core-distance* and *rechability-distance* in order to easily see the effect of a lower ϵ:

- The *core-distance* of an object p, $core(p)$, is the smallest distance such that the $core(p) - neighborhood$ of p contains exactly $MinPts$ objects (for objects that are non-core w.r.t. ϵ, this value is undefined)
- The *reachability-distance* of an object p w.r.t. another object o, $reach(o, p)$, is the smallest distance such that p is within $reach(o, p) - neighborhood$ of o and o remains a core object w.r.t. $reach(o, p)$ (for objects that are non-core w.r.t. ϵ, this value is undefined)

Storing this information, OPTICS is able to extract all density-based clusters w.r.t. any distance ϵ' that is smaller than the distance ϵ used in generating the order. The result yield by OPTICS is an ordering of the objects $o : \{1..n\} \rightarrow DB$ and the corresponding reachability-values $r : \{1..n\} \rightarrow \Re_{\geq 0}$.

An example of OPTICS output is reported in Fig. 2.9.

An important observation is that both DBSCAN and OPTICS extensively perform neighborhood calculations, needing for a great deal of spatial queries on the database. Thus, to efficiently perform such queries, they rely on spatial index structures such as $R^* - tree$ or $X - tree$. However, these spatial indexes perform well on low-dimensional data while their efficiency decreases as the number of dimensions increases. This limitation has led Hinneburg and Keim to propose *DENCLUE* (*DENsity-based CLUstEring*) [HK98], a clustering algorithm based on the following ideas:

- each spatial object has an influence on its neighbors, that is formally modelled by a mathematical function called *influence function*;

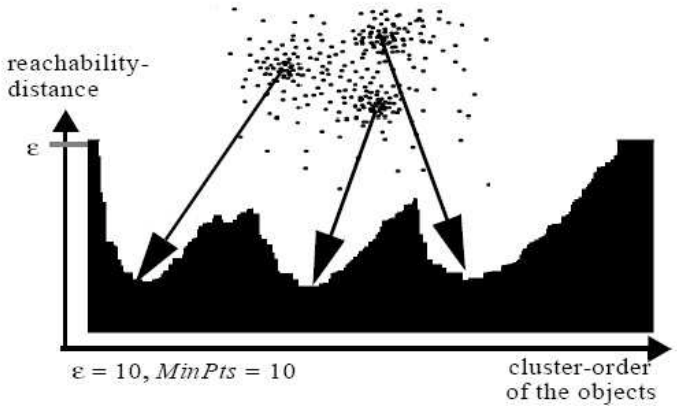


Figure 2.9: An example of OPTICS output plot

- the overall density in a given object can be analytically modelled as the sum of the influence functions of all data points over it;
- clusters can be determined by identifying *density actractors*, that is, local maxima of the overall density function.

DENCLUE moves from the concept of dense areas, conceived as areas with an high number of objects arranged in a limited space, to the concept of areas where the influence of other points exceeds a fixed threshold. In order to improve its efficiency, DENCLUE summarizes the influence values of the objects in a grid-like structure. Anyway, it cannot be considered to belong to grid-based methods (described in the following subsection) since it uses a grid structure only to make calculation of density function more efficient and not as a basis of its computations.

2.2.4 Grid-based methods

A drawback of density-based methods is that their efficiency decreases as the number of data dimensions increases. *Grid-based methods* overcome this limitation by partitioning data to be clustered in a grid structure. In particular, data are quantized into a finite number of cells that represent the elements that will be clustered. In this way, the clustering process becomes fast in comparison with other approaches because clustering is performed on cells rather than data objects, even if some quality loss in resulting clusters is observed depending on the granularity of grid in which data are summarized.

An example of a grid-based clustering method is *STING* (*STatistical INformation Grid*) [WYM97]. It divides the dataset into a grid of rectangular cells. Such grid is calculated at different levels of granularity, originating a hierarchical structure where at each level corresponds a different level of resolution; each cell at a given level in the hierarchy is partitioned to form a number (usually four) of cells at the next lower level. An example of such iterative subdivision is depicted in Fig. 2.10.

Each grid cell is described in terms of information concerning the summarized

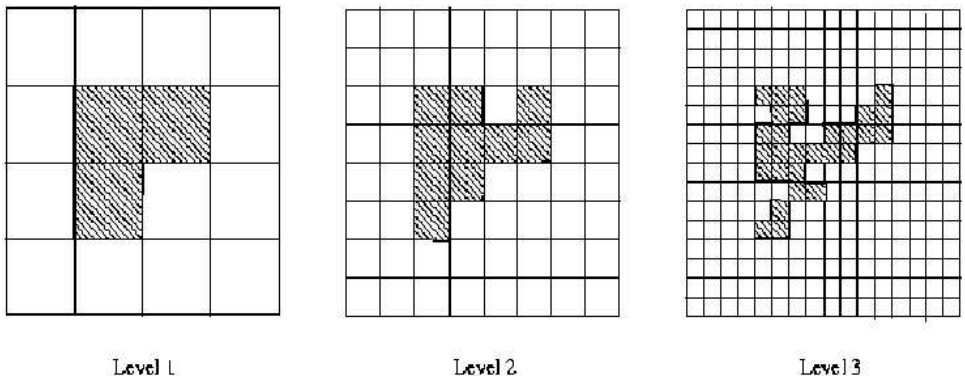


Figure 2.10: Three consecutive layers of a STING structure

objects, namely the number of objects, their mean, standard deviation, minimum, maximum and the type of distribution characterizing the attribute value in the cell (normal, uniform, exponential or none). The method performs only one scan of the database to generate the bottom-most level cells, that is, cells with the higher resolution, while values of upper levels cells are calculated from their children cells in the lower level. Clustering is performed by using a top-down approach, starting from a user-specified density-level (typically, with a small number of cells). For each cell in the current layer, the confidence interval is computed that estimates how much the cell is relevant to the result of the clustering. Cells that are deemed irrelevant are cut out, while relevant cells are further examined at the next level (and at a greater level of details). This process is repeated until the bottom layer is reached. After the bottom level of structure is reached, relevant cells of the bottom level are combined into relevant regions (based on grid-neighborhood) and resulting regions are returned as detected clusters.

STING presents several advantages: first, the grid-based computation is query-independent, because information stored in each cell and summarizing objects in the cell is independent of the query; second, the grid structure can be easily elaborated and can be incrementally updated; third, this method is very efficient, since the time complexity of the grid generation step is $O(n)$, where n is the number of objects in the database, while the query processing time is $O(g)$, where g is the number of cells in the bottom level of hierarchy (with $g \ll n$).

On the other hand, the quality of detected clusters depends on the granularity of the cells in the lowest level of the structure: if such granularity is too fine, the computational cost of the algorithm substantially increases, while if it is too coarse, the quality of results decreases. In addition, the detected clusters can have only horizontal or vertical boundaries, while diagonal boundaries cannot be detected.

A method combining grid-based and density-based approaches in cluster detection is *WaveCluster* [SCZ98], that uses a *wavelet transformation*, that is a signal processing technique that decomposes a signal into different frequency subbands, to transform the original feature space into another one where dense regions are searched. Since the method works at different level of resolutions, it can discover

clusters at varying levels of accuracy. In addition, it results to be very efficient, with a computational complexity equal to $O(n)$, and does not require any input parameter. These characteristics make WaveCluster a very attractive method.

A combination of grid-based and density-based approaches is also adopted by *CLIQUE* (*Clustering High-Dimensional Space*) algorithm [AGGR98], that is able to work with high-dimensional data in large database. It generates non-overlapping rectangular units representing a partition of the d-dimensional data space and detects dense units, that is, units having a number of objects greater than an input parameter. Dense units in a high-dimensional space are detected by resorting to the *apriori property* used in association rule mining: *If a k-dimensional unit is dense, then so are its projections in (k-1)-dimensional space*. Inverting this rule, if we detect some non-dense units in (k-1)-dimension, then we can state that the k-dimensional unit cannot be dense.

By following this approach, the detection of dense units in lower dimensional spaces can heavily restrict the areas to be analysed in high-dimensional space. Dense units are then examined to determine the clusters. Finally, for each detected cluster, CLIQUE determines the maximal region that covers the cluster of connected dense units. The strengths of this method is that it is insensitive to the order in which objects are read and does not assume any distribution in data. Furthermore, it scales well with the increase of both the size of input data and the number of dimensions in data.

2.3 Graph-based clustering methods

In the context of spatial clustering, a natural and very understandable way to represent the spatial relations among objects is represented by graphs. Indeed, by representing each spatial object with a node and drawing an edge between two nodes if and only if a (spatial) relation exists between the corresponding objects, we can easily represent all the input data by means of a graph structure. For instance, if we consider as spatial relation the Euclidean distance, we can obtain the graphs mentioned at the end of subsection 2.1.1.

Due to their high expressive power, graph-based data representations have been used in a number of data mining works. In [GHC01] some settings are presented where different forms of concept learning are performed using graphs. In particular, an extension of Subdue data mining tool [JCH01] is presented that is able to learn concepts in term of graph substructures from data composed by positive and negative examples in graph format. A similar form of supervised learning relying on positive and negative examples in graphs has been outlined by Holder and Cook [HC03], where supervised learning and graph grammar induction are considered as important extensions of *GBRL* (*Graph-Based Relational Learning*) field. Here, both subfields aims at extracting knowledge from data described in form of graphs yielding hypotheses in form of subgraphs or grammars, respectively. However, all these works in GBRL have to face with some open issues, such as the high computational

cost and the difficulties to process very large graphs (scalability).

In recent years, the possibility of combining graph-based learning studies in clustering tasks has been investigated by an increasing number of researchers. An important contribution in such direction has been provided by [JCH01], where Subdue has been presented as a novel hierarchical conceptual clustering method based on graphs. In particular, this method aims at mining models expressed as substructures frequently occurring in the given graph. Each instance of the mined substructure is replaced by a single node representing the whole induced model allowing a form of graph compression, and the mining process is iteratively applied on the new graph. Here, each mined substructure represents a cluster and each instance of the substructure in the graph represents an item belonging to the cluster. Since the final result is a set of substructures possibly containing nodes representing lower level substructures, this form of learning is considered a form of hierarchical clustering. However, this approach strongly relies on recurrent link among different instances of the same nodes, but is not able to group together similar linked items in graphs. In [NAJ03] the authors remark that data are characterized by both the descriptive attribute information of objects and the structure of relationships relating different objects. Two different approaches to clustering problem stem from this observation: data clustering methods, that group together objects on the basis of their attributes similarity, and graph partitioning methods, that aim at grouping objects according to the strength of their relationships. Since the above features are two different aspects of the same data, the authors suggest to merge both the approaches into hybrid methods in order to achieve better results as concerns cluster quality and noise tolerance. To this aim, they introduce the concept of *community clusters*, that is, groups of items having similar attributes and also being highly inter-connected.

As concerns the spatial clustering environment, graph-based data representation has been used by *HPGCL* (*Hierarchical Parameter-free Graph CLustering*) algorithm [And03]. This is an agglomerative clustering method that computes objects similarity on the basis of the computation of a set of neighbourhood graphs. The strength of this method is that it does not need neither parameters like thresholds nor assumptions about the distribution of the data or number of clusters. On the other hand, the homogeneity evaluation is computed by only taking into account links (or relations) among objects, without caring about the likenesses among clustered objects. Differently from previous approaches, in CLARANS [NH94] clustering is performed as a search through a graph where each node represents a particular configuration of k-medoids and two nodes are connected if their sets differ by only one object. Although a graph structure has been used to detect clusters, in this work a graph represents a sort of hypothesis space to search rather than arrangement of spatial objects. For this reason, the importance of graph structures is considered to be secondary, and most of researchers prefer to classify CLARANS as a partitioning method rather than a graph-based method.

2.4 Conclusions

In this chapter, we have introduced the most important issues that must be faced when dealing with spatial data. Indeed, spatial data are characterized by both a descriptive nature and some spatial features (such as shape and position). This makes inadequate the traditional clustering methods, since they can only detect non-spatial similarities among objects in domains where items should be grouped according to their spatial locality as well.

The necessity to deal with an ever increasing amount of spatial data has led to the development of a substantial number of spatial methods belonging to different approaches. Most of approaches proposed in literature and their most representative methods have been illustrated in Section 2.2.

Spatial environment can be considered as composed by spatial objects related each others by implicit spatial relations. This suggests to arrange data into a graph structure, where nodes correspond to objects and edges describe existing relations. Methods that performs some form of learning on graphs are called *graph-based* and have been discussed in Section 2.3.

However, the main observation arising from the cited works in graph-based clustering is that a gap exists between the graph-partitioning algorithms and the data clustering ones. The former ones tend to put together objects strongly connected each other or consider clusters as frequent substructures in graphs, while the latter ones evaluates objects similarity taking only into account non-spatial attributes.

Finally, both spatial and graph-based clustering methods assume that spatial data are described in terms of features concerning the entire area, neglecting that often in spatial environments data are complex structures composed by sub-elements and relations among them. This means that existing spatial methods usually make strong and restrictive assumptions on data representations and result to be unable to deal with structured data. Resorting to multi-relational data mining could represent a good solution to improve the clustering results.

Chapter 3

Multi-Relational Data Mining and Inductive Logic Programming

In the previous chapter we have dealt with data mining methods designed for pattern discovery over data. However, all the approaches and methods discussed till now are characterized by a strong assumption over data: they assume that data are stored in a single table and each observation corresponds to one tuple. It is easy to see how this assumption is very restrictive since data are generally represented in relational databases by means of multiple tables related each others. This makes classical data mining approaches inadequate to deal with real structured data unless strong simplifications are performed over data description that reduce their predictive or descriptive capacity. Thus, a different approach has been proposed in literature, namely (multi)-relational data mining that, representing data in a more natural way, is able to discover more expressive and useful patterns.

This chapter, and the next section in particular, investigates on limitations of classical data mining methods and discusses the motivations that gave rise to multi-relational approaches. Since most of DM algorithms have been developed to work in the propositional environment while data are in general expressed in relational format, two general methodologies will be presented in Section 3.2: one of them aims to turn a propositional system into a relational one, while the other one transform relational data into a propositional format. Section 3.3 will show how ILP can be adopted to pursue the aims of relational data mining and the main concepts that characterize ILP learning problems. In order to present an example of ILP learning system for the induction of relational models, a system named ATRE and its main peculiarities will be described in Section 3.4, while Section 3.4.1 will be devoted to show how ATRE can be used to learn models from only positive examples. Section 3.5 will present a particular family of clustering methods where most of studies in relational data mining has been dedicated: conceptual clustering. Concluding remarks will be drawn at the end of this chapter.

3.1 The Multi-Relational approach

In the previous chapter we have discussed a great deal of methods belonging to different approaches and conceiving clustering in several fashions. All the presented approaches look for patterns in data stored in the traditional matrix form: rows represent observations and columns represent features of data. This way of conceive data is fully compliant with the way in which databases arrange them in tables: each tuple correspond to an observation, while attributes represent features describing an observation. This form of representation is known as *propositional* or *attribute-value representation*, since each feature (or attribute) is described with only a single, primitive value. For instance, if we consider data stored in a table listing selling information of a car vendor, an observation could be represented as follows:

$$young\_buyer = yes, car\_type = sporting, car\_price = 12500,$$
$$mp3\_reader\_equipped = yes, color = red, doors = 3, \ldots$$

It is easy to see how examples expressed in this formalism can be straightforwardly mapped into a single table of a relational database. Here, a possible mined rule for the characterization of young buyers could be:

$$young\_buyer = yes \leftarrow (car\_type = sporting) \wedge (car\_price \leq 15000) \wedge$$
$$(mp3\_reader\_equipped = yes)$$

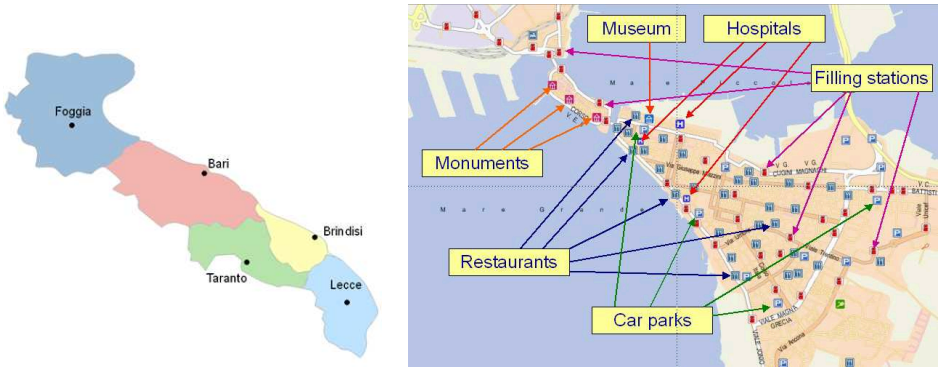
stating that youngs typically buy sporting cars, which cost less than 15000 Euros and equipped with a mp3 reader.

In this representation, the underlying assumption is that observations are "flat", in the sense that they are composed by a fixed number of features, with a rigid structure and with all the information needed for analysis encoded in only one table. Such assumption is known in literature as *single-table assumption* [Wro01].

By adopting this representation formalism it is possible to develop efficient data mining algorithms. However, data about the real world are seldom of this form. Data to be analysed (*units of analysis*), even if concerning instances of the same concept, are typically composed by a variable number of entities (*units of observation*), often of different types and related each others. Consequently, in databases data are scattered over different tables (relations), each of them with a particular structure and linked each other by a chain of dependencies.

This is particularly true in spatial environments, where data to be analysed are often composed by sub-units heterogeneous and related each others. For instance, suppose we want to cluster cities in Apulia (Italy) according their structural similarity. Spatial data are represented in Figure 3.1.

From Figure 3.1 it is clear that each city to be clustered (unit of analysis) is composed by a large number of sub-units (units of observation) describing hospitals, restaurants, monuments, filling stations, etc., that are heterogeneous each others and, hence, cannot be represented in the same table. In this case, a different table should be used to store each different type of unit of observation: a table for the hospitals, a table for the restaurants, and so on. Furthermore, topological as well as other spatial relations relate these units of observations and define the structure of the spatial object in question, representing the city as a whole.



(a) Cities to cluster (b) Details on Taranto

FIGURE 3.1: An example of structured objects in spatial data

As another example, if we want to perform some form of DM analysis on information gathered by a generic trade company, we are likely to deal with a relational database containing tables similar to the tables 3.1.

CUSTOMERS

ID	Name	Address	City	Sex	Age	Social Status	Trustworthy
p1	Smith	61, Lake s	London	m	37	single	no
p2	Brown	108, Central Park	Bristol	f	44	married	no
p3	Palmer	98, Shakespeare ave	London	f	31	single	yes
...	...	...	...	...	...	...	...

ORDERS

Order ID	Customer ID	Quantity	Cost	Deferred Payment	Instalments	...
o85	p1	52	5200	yes	50	...
o104	p2	20	2200	no	-	...
o136	p2	35	3000	yes	60	...
o155	p3	32	3200	yes	20	...
...	...	...	...	...	...	...

TABLE 3.1: An example of Multi-relational data

In the first table information about the customers are stored; the identifiers (ID) are used to uniquely detect each customer, while the other fields (Name, Address, Sex, Age, etc.) represent the descriptive attributes relevant for the domain we are dealing with. Since it represent each customer with a single tuple, it can straightforwardly represented in an attribute-value formalism. The second table stores information about orders placed by customers, holding details pertaining each order in a single tuple. Here a one-to-many relation holds between the customers table and the orders table, since each customer can place one or more orders. This

relation is determined by means of the `CustomerID` field in the `Orders` table.

In general, in database environment, a field in a relation (`Orders.CustomerID`, in our example) that points to a key field in another relation (`Customers.ID`) is called a "foreign key", indicating that the value in this field is required to be a key value in the other relation.

However, it should be always kept in mind that, even if important analogies exist between the representation formalisms adopted by DM and DataBase Management Systems (DBMS), the purposes of these two disciplines are different: database environments are responsible for the storage and manipulation (insertion, update, delete, modification) of data, while DM aims at analyse them to extract patterns.

Taking into account the example of table 3.1, suppose we want to apply DM analysis to characterize untrustworthy customers. By examining only one table at a time we can not mine any useful pattern, so classical DM approaches relying on single-table assumption would fail on this task. In order to represent orders information in the customers table, one could merge the orders table into the customers one by performing a left outer join, but we could only consider once-only customers. In fact, in propositional approaches each observation (a customer, in this case) can be represented in one tuple, so neither we can have more than one tuple referring to a particular customer, nor a table cell (corresponding to a given attribute of a particular observation) can contain more than one value.

One could try to summarize orders information by means of aggregate operators (for instance, by including in the customers fields the sum or average quantity and total cost over orders) but this clearly implies an information loss. In fact, units of observation describe objects sometime of different nature, therefore units of analysis cannot always be constructed by simply aggregating (i.e. min, max, count or average) properties of the corresponding units of observation. Conversely, it may be important to distinguish units of observation which represent target objects of analysis from other target-relevant objects and represent the relationships among them. Modeling properties of these different objects as well as relationships among them is a key challenge both in descriptive and predictive problems that arise in complex domains, such as spatial domains [SSV+02] or biological domains [DBK+99], where the implicit description or the prediction of a property of a target object can be strongly affected by properties of target-relevant objects according to the relationships among them.

Furthermore, one could try to expand the customers table by adding as many fields as requested to include all necessary information into a single tuple. This would lead to add the fields $OrderID_n$, $Quantity_n$, $Cost_n$, $DeferredPayment_n$ and $Instalments_n$, with n sufficiently high. It is clear that this approach cannot be applied in general since it would imply an unnatural way of represent data. Putting one of the related data in the column "$Field_1$", another one in "$Field_2$", and so on, would assign a numbering to the fields that could not correspond to reality, but assigned arbitrarily.

Moreover, observations of some domains are even impossible to be fully represented. In fact, suppose we want to apply DM techniques to characterize people suffering from diabetes. A possible representation of observations could be:

PATIENTS

ID	Job	Obese	Sex	Age	Social Status	Diabetes
...	...	...	...	...	...	...
p15	countryman	no	m	86	married	yes
p34	countryman	no	f	59	married	no
p65	clerk	yes	m	43	married	no
p66	teacher	no	f	35	single	no
p67	miner	no	m	31	single	yes
p82	teacher	yes	f	37	married	yes
...	...	...	...	...	...	...

PARENT

Child ID	Parent ID
...	...
p34	p15
p65	p23
p65	p18
p67	p34
p82	p34
...	...

Table 3.2: An example of recursion in data

By examining tables 3.2, one could observe that two out of three cases of diabetes are due to heredity: in fact, $p67$ and $p82$ are sons of $p34$, which does not suffer from diabetes, but is son of $p15$ that is a diabetic. The simplest and more expressive way of characterize diabetics is by means of recursion:

$ancestor(X, Y) \leftarrow parent(X, Y).$

$ancestor(X, Y) \leftarrow ancestor(X, Z) \wedge parent(Z, Y).$

$diabetic(X) \leftarrow ancestor(Y, X) \wedge diabetic(Y)$

but recursion cannot be represented in propositional settings.

Thus, the analysis methods working with propositional representations will have no chance to discover the right model. *Relational Data Mining* (RDM), also referred as *Multi-Relational Data Mining* (MRDM) to emphasize the fact that it deals with many relations over data, represents the answer of DM to the limitations highlighted above. It resorts to *Inductive Logic Programming* (ILP) to easily model structured data and relations among different entities with the same representation formalism. In addition, ILP results to be surprisingly effective in dealing with recursion definitions and to take into account *Background Knowledge*, that collects general rules and facts holding in the considered domain. This makes MRDM (and ILP) an attractive tool for mining pattern from data, even if we have to pay an higher computational complexity due to a more expressive power. Further details on ILP will be provided in Section 3.3.

3.1.1 Some MRDM algorithms

Up to now, work in ILP has mostly concentrated on *conceptual clustering*, a clustering approach that aims at detecting an intensional description of each cluster in the given representation language other than the list of objects belonging to it. Further details on this approach will be provided in Section 3.5.

However, other than algorithms belonging to conceptual clustering, other methods have been developed by relying on the more powerful expressiveness of RDM. Some of these methods are RIBL and RDBC.

RIBL (*Relational Instance-Based Learning*) [EW96] is an instance-based learning algorithm that defines an improved similarity function in order to be able to compare first-order descriptions. An *instance-based algorithm* does not compute a generalization of the learning examples at learning time, but simply stores the examples and then performs classification by comparing a new instance to the previously stored instances. In particular, predictions are based on those k examples from the set of examples that are closest to the observation to predict, where k is a user-given or system determined parameter, in the sense that the new observation is assigned the most frequent value occurring in the k nearest neighbors (algorithms working in this way are called in literature *k-nearest neighbor classifiers*). Since generalization is delayed until classification time, such algorithms are also referred to as *lazy learning* algorithms. RIBL turns the basic instance-based learning problem into a first-order version by allowing each instance to be described not only by numerical and discrete attributes, but also by attributes of type `object` containing

object identifiers that point to further information in a separately specified body of background knowledge, that is a database consisting of first-order ground atoms (first-order facts). Moving from propositional to first-order representations involves a further degree of complexity in the similarity calculation, since now instance descriptions not only contain attribute values, but also references to background knowledge. To this end, RIBL uses a recursive descent strategy that, while evaluating the similarity of two descriptions, treats the corresponding atoms as instances to be compared, until all comparisons are reduced to the propositional case.

However, RIBL was limited to consider function-free relational representation, so Horváth et al. [HWB01] have extended RIBL and overcome such limitation by also handling list and other functional terms. Comparisons of lists and terms are performed by relying on the idea of *edit distances* between objects. Intuitively, with an edit distance, we are given a set of edit operations (insert, delete, or change) with an associated cost function that tells us how expensive it is to delete or insert an element of one of the compared objects, or how expensive it is to change one into another. The edit distance is given by the cheapest (i.e., smallest cost) sequence of such operations that turns the first object into the second.

More formally, the learning problem solved by the resulting version of RIBL can be described as follows:
Given:

- a set of examples E, each described by a set of attributes $\{A_1, \ldots, A_n\}$ of type `number`, `discrete`, `object`, `constant`, `list` or `term`
- a target value $c(e)$ for every $e \in E$
- background knowledge B consisting of first-order atoms with arguments of type `number`, `discrete`, `object`, `list` or `term`
- a set of unseen instances E_{new} each described by the same attributes as E

Find: a partition $c'(e)$ of the target value for each $e \in E_{new}$.

RDBC (*Relational Distance-Based Clustering*) [KW98] is an agglomerative hierarchical clustering method for observations represented in the first-order logic formalism. It starts out with the list of cases as one-element clusters, and iteratively groups the two most similar clusters into a new cluster until eventually all the items are grouped in a single cluster. Since the classical distance measures are not suited to deal with first-order descriptions, RDBC relies on a similarity definition based on that one adopted by RIBL:

$$sim(C_i, C_j) = \frac{1}{n_{C_i} n_{C_j}} \sum_{k=1}^{n_{C_i}} \sum_{l=1}^{n_{C_j}} sim_{RIBL}(o_k, o_l)$$

where: n_C is the number of items belonging to the cluster C, o_k belongs to C_i, o_l belongs to C_j, and $sim_{RIBL}(o_k, o_l)$ denotes the similarity function inherited from RIBL.

Starting from the observation that similarity between clusters decreases as we move from the leaves towards the root, this method uses a search procedure that

prunes the clustering tree according to growing intra-cluster similarity thresholds, and selects the threshold and corresponding group of clusters that maximizes the average intra-cluster similarity.

Finally, Batagelj and Ferligoj in [BF00] consider two special cases of the general clustering of relational data problem, namely *clustering with relational constraints* and *blockmodeling problem*. The former problem concerns particular clustering tasks where the goal is to group similar units also satisfying additional conditions over data. Since the geographical contiguity is a special case of relational contraint, this approach can be applied to cluster similar geographical regions such that the regions inside each cluster are also geographically connected. The latter problem considers data as a network composed by a set of units and one or more binary relations on it. Blockmodeling, that has particularly adopted in social network analysis, seeks to cluster units that have substantially similar patterns of relationships with others. Its basic goal is to reduce a large, potentially incoherent network to a smaller comprehensible structure that can be interpreted more readily.

3.2 From propositional to relational

As stated earlier in this chapter, most of existing DM methods have been developed to work with a propositional (or attribute-value) representation. On the other hand, the widespread adoption of relational (or first-order) representation for data, due both to an ever increasing level of details in gathered data and the success of relational databases, makes the extraction of relational patterns a necessity.

In order to fill the gap between existing DM algorithms and the additional expressivity of the relational data representation, two possible methodologies can be adopted:

- upgrading propositional learners to first order logic
- propositionalization of relational data

The first methodology aims at upgrading existing attribute-value learners towards first-order logic. By extending existing methods involves some advantages: firstly, first-order version of the systems are easier to understand and use by users already familiar with the propositional case. Secondly, starting from an already existing system, it is possible to exploit the expertise and heuristics well-established in the propositional environment. Thirdly, results for the propositional data (that can be considered as a particular case of first-order data) are already available to validate the system.

From the knowledge representation point of view, upgrading the propositional formalism towards to first-order allows to solve some key problems: the number of elements composing an observation and the corresponding relations among them does not need to be fixed in advance and observations can be represented without imposing an unnatural ordering of the features. In addition, differently from the propositional environment, the relational representation of data allows to take into

account rules holding in the working domain in addition to factual knowledge in the examples. These rules, that are common to all the examples, are referred as *Background Knowledge* (KB). As BK is visible for each example, all the facts that can be derived from it and a given example are part of the extended example, that is the *minimal Herbrand model* of the example and BK (see Section 3.3).

Even if different strategies can be adopted to turn an existing propositional system into a relational one, a general conversion methodology can be represented by the following steps [VD01]:

Step 1:	Identify the propositional learner that best matches the learning task
Step 2:	Use interpretations to represent examples
Step 3:	Upgrade the representation of propositional hypotheses by replacing attribute-value tests by first-order literals and modify the coverage test accordingly
Step 4:	Use θ-subsumption as the framework for generality
Step 5:	Use an operator under θ-subsumption. Use that one that closely corresponds to the propositional operator
Step 6:	Use a declarative bias mechanism to limit the search-space
Step 7:	Implement
Step 8:	Evaluate your (first-order) implementation on propositional and relational data
Step 9:	Add interesting extra features

Table 3.3: An overview of the methodology for upgrading propositional learners to first-order logic

The first step is straightforward. The second one suggests to upgrade the propositional representation of examples to a first-order one. This can be done by constructing a fact of type $example(val_1, \ldots, val_n)$ for each tuple of the table *example* having val_i as the value of the i-th attribute. Alternatively, each tuple (observation) can be represented by the set of facts $\{att_1(val_1), \ldots, att_n(val_n)\}$.

As concerns the third step, generated hypotheses can be syntactically upgraded to a first-order representation by replacing the attribute-value tests in the rules with literals having one or more arguments. For instance, the propositional test $color = red$ can be replaced by the literal $color(X) = red$. Anyway, we also need to specify when an example is covered by a hypothesis. Thus, an example e will be covered by a hypothesis H (or a set of rules defining the $class(c)$) if $H \wedge e \models class(c)$.

In order to search the space of hypotheses, machine learning systems structure the search-space by means of "*is more general than*" relation, that imposes an ordering among the generated rules. In propositional representations such relation is generally quite simple: rule A is considered more general than rule B if the literals occurring in A represent a subset of those occurring in B. In first-order representations, the generality relation is more complex and different tests have been proposed in literature to estimate it: θ-subsumption, inverse implication, inverse resolution and inverse entailment are some of them [MD94]. The most used is θ-

subsumption that, other than being not very computationally expensive to adopt, works at the level of single rules instead of sets of rules, similarly to propositional systems. θ-subsumption, however, is not suited to be adopted in the recursive theories or multiple predicates learning [Mal03].

In the fifth step, we are required to define operators for searching the hypothesis space according to the generality order adopted. Typical operators are:

- *specialization* (or *refinement*) *operator* that, given a rule, generates a less general rule by adding a literal or turning a variable into a constant
- *generalization operator* that, conversely, generates a more general rule by removing a literal from a given rule or turning a constant into a variable
- computation of the *least general generalizzation* (lgg) of two rules, where $lgg(A, B)$, with A and B rules, is a rule that θ-subsumes both A and B, and for every rule G θ-subsuming A and B, G also θ-subsumes $lgg(A, B)$

Since the hypothesis space to explore can be wide and it could be intractable to completely explore it, after an operator has been defined for the search of rules in the hypothesis space we need to constrain the search space by specifying some form of declarative bias that limits the number of rule considered. This can be done in different ways: by bounding the number of variables or literals in the rules or by adopting some form of syntactic limitations. The remaining steps are self-explanatory.

The second methodology follows a reverse path to match algorithms and data representations. Instead of implementing a relational version of existing propositional DM approaches, it transforms relational problems into a format suitable for propositional learning algorithms. This conversion process is called *propositionalization* [KLF01]. However, such conversion is not costless, but requires some issues to be addressed. Firstly, in ILP data are structured, that is, they are composed by a set of sub-elements possibly heterogeneous. Since the goal is to put all this information into a single table, this could lead to span a given observation along different tuples. In this case, the resulting framework is not propositional because it infringes the requirement that an observation must be stored in one tuple. Thus, we exclude from this study transformation approaches which result in a table where several rows correspond to a single example. Propositionalization through first-order feature construction can mostly be applied in so-called *individual-centered* domains, where there is a clear notion of individual and learning occurs on the level of individuals only. As a consequence, since individuals are represented by a single variable, the predicates whose definition is to be learned are either unary predicates concerning a boolean property of individuals, or binary predicates assigning a class value to each individual. For instance, a family domain, where the goal is to learn the concept of $daughter(X, Y)$, is not an individual-centered domain because the concept to be learned (*target concept*) cannot be defined without referring to two different individuals.

Secondly, BK is extensively used in first-order settings to encode general knowledge (both in form of rules and facts) assumed to be true in the working domain, but it does not exist in propositional settings. BK is one of the strenghts of ILP with respect to propositional approach, since it may substantially improve the results of learning in terms of accuracy, efficiency and explanatory power of induced hypotheses. So, how can we represent information embedded in BK in propositional approaches? The only way to solve this problem is to define new features appositely aimed at representing this additional information. This leads to the concept of *constructive induction*, that highlights the ability of a learning system of creating new predicates from the existing ones and that has its ILP equivalent in *predicate invention*. Analogously, constructive induction is employed to code structural properties, that is properties relating the different tables storing information of an observation. This leads to another drawback of propositional representation: the number of features to be considered in propositional case exponentially grows as either the number of tables representing structural information or the number of BK rules increase.

From the above discussion, it is clear that both the presented approaches represent a straining in the process of matching data representation and available DM algorithms. The upgrade of propositional learners to first order logic is the most expensive process in that it requires the implementation of another version of the learning systems, but avoids to perform data conversions that can imply information loss. On the other hand, the propositionalization of relational data does not require any change in the available DM algorithms, but involves a complex manipulation of available information, since data are first converted into a propositional format, analysed with DM algorithms, and then re-converted into relational format.

3.3 Data Mining and ILP

The above sections have demonstrated how, thanks to Inductive Logic Programming (ILP), MRDM is a cut above with respect to classical DM approaches in discovering more useful and expressive patterns that also take into account the structure of data [DL01]. In addition, ILP is able to take into account generally valid background domain knowledge (BK) while detecting patterns. Furthermore, ILP adopts the same representation formalism to describe data, BK and discovered patterns; this allows us to re-use discovered patterns as BK in the subsequent learning tasks.
More specifically, *Inductive Logic Programming* (ILP) is a research area at the intersection of machine learning and logic programming. It aims at define a formal framework and practical algorithms for inductively learning from examples relational descriptions expressed in a subset of first order logic representation formalism, that is, to perform *specific to general reasoning* [LD94].

From inductive machine learning, ILP inherits its goal: to develop tools and techniques to induce hypotheses from observations (examples) and to synthesise new knowledge from experience [MD94].

From computational logic, ILP inherits its representational formalism, that is used to represent both observations and hypotheses. In this way, two main limitation of classical machine learning techniques can be overcome:

1. the use of the limited knowledge representation formalism of the propositional logic
2. difficulties in using background knowledge in the learning process

The first limitation is important because many domain of expertise cannot be expressed in the propositional logic. As concerns the second limitation, the use of domain knowledge is proven to be essential for achieving intelligent behaviour. Logic offers an elegant formalism to represent knowledge and hence incorporate it in the induction task.

In this section some basic notions on first order logic formalism and ILP setting will be presented, in order to facilitate the understanding of the learning aspects discussed in this dissertation.

First of all, a *first-order logic language* $\mathcal{L}$ is defined by an alphabet composed by seven sets of symbols: variables (starting with a capital letter), constants (starting with a small letter), function symbols (e.g., $f(\ldots)$, $g(\ldots)$), predicate symbols (e.g., $p(\ldots)$, $q(\ldots)$), logical connectives ($\neg$, $\wedge$, $\vee$, $\leftarrow$, $\leftrightarrow$), quantifiers ($\exists$, $\forall$) and punctuation symbols ("(", ")" and ",").

A *term* can be defined as follows:

- a variable is a term
- a constant is a term
- $f(t_1, \ldots, t_n)$ is a term, where f is a function symbol with arity n and $t_1, \ldots, t_n$ are terms

An *atomic formula* or *atom* $p(t_1, \ldots, t_n)$ is the application of a predicate symbol p with arity n to n terms. A *literal* is an atom (*positive literal*) or its negation (*negative literal*). A formula is said to be a *well-formed formula* (wff) if it is syntactically correct and is inductively defined by either an *atomic formula* or by combining atomic formulas by means of logical connectives and quantifiers.

Concerning the quantifiers, the *scope* of a quantifier $\forall X$ (resp. $\exists X$) in $\forall X : F$ (resp. $\exists X : F$) is F; each occurrence of X in F is said to be *bound*. Any other variable in F that does not immediately follows the quantifier is said to be *free*. If a formula does not contain free variables it is a *closed formula* (or *sentence*), otherwise it is called *open formula*. A *variant* of a formula is another formula obtained by renaming all its variables.

A special type of formula is a *clause*, that is a disjunction of literals of the form:

$$\forall X_1 \forall X_2 \ldots \forall X_s (A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m)$$

where A_i, B_i are atoms and X_1, X_2, ..., X_s are all the variables occurring in the formula. The clause above mentioned is typically represented in the following way:

$$A_1; \ldots; A_n \leftarrow B_1, \ldots, B_m$$

The part on the left of the symbol $\leftarrow$ is called *head* (or *subsequent*), while the part on the right of the symbol $\leftarrow$ is called *body* (or *antecedent*) of the clause.

If the clause does not contain negative literals ($m = 0$), the clause is called *fact* and usually is represented without $\leftarrow$ symbol. Conversely, if the clause does not contain positive literals ($n = 0$), it is called *denial*. A clause having at most one positive literal ($n \leq 1$) is called *Horn clause*, while a clause having exactly one literal in the head ($n = 1$) is said to be a *definite clause*. A clause is said to be *range-restricted* if the variables appearing in the head are a subset of those appearing in the body. A *connected* clause is a clause where each literal has at least one argument in common with another literal in the clause. A *ground clause* (resp. term) is a clause (resp. term) without variables. A *(definite) logic program* is a set (or a conjunction) of (definite) clauses. A *Datalog clause* (resp. program) is a definite range-restricted clause (resp. program) that does not contain function symbols except variables. Since datalog programs cannot contain functions as clause arguments, they are so suited to interact with large DBs that Datalog is considered to be a database query language based on the logic programming paradigm [CGT89].

A substitution $\theta = \{X_1/t_1, \ldots X_k/t_k\}$ is a function mapping variables X_i to terms t_i. Applying a substitution θ to a clause C means to replace all the occurrences of each variable X_i in C with the corresponding term t_i.

Up to now, we have only dealt with syntactic aspects of first order logic, but it remains useless if we cannot associate it with a meaning. The semantics of a set of formulas is defined in terms of interpretations and models.

In general, an *interpretation* is an assignment of truth values to constants and predicate symbols. Alternatively, an interpretation can be considered as the set of ground facts considered to be true in the working domain. For instance, an interpretation $\mathcal{I}$ can consider to be true the predicates $father(burt, mary)$ and $mother(sophie, mary)$ and false the predicates $father(mary, burt)$ and $mother(mary, sophie)$, so $\mathcal{I} = \{father(burt, mary), mother(sophie, mary)\}$.

If a clause C is true under the interpretation $\mathcal{I}$, we say that $\mathcal{I}$ *satisfies* C, or that $\mathcal{I}$ is a *model* for C, notation $\mathcal{I} \models C$. A set of clauses S is *satisfiable* if at least one interpretation $\mathcal{I}$ exists such that $\mathcal{I} \models S$; otherwise it is said to be *unsatisfiable*. Moreover, a clause F *follows logically* from (or that it is a *logical consequence* of) a set of clauses S or, alternatively, that S *logically implies* (or *logically entails*) F, iff each interpretation satisfying every clause in S also satisfies F. In this case, we write $S \models F$.

However, in order to have a reasoning tool easy to implement, we focus our attention on interpretations of a particular type, called *Herbrand interpretations*. The *Herbrand universe* $\mathcal{H}$ of a language $\mathcal{L}$ or a program is the set of all the ground terms that can be obtained by combining the available symbols. The *Herbrand base* $\mathcal{B}$ of a language $\mathcal{L}$ or a program is the set of ground atoms. A *Herbrand interpretation* $\mathcal{I}$ is a subset of the Herbrand base ($\mathcal{I} \subseteq \mathcal{B}$).

Herbrand models are characterized by an important property: the intersection of a set of Herbrand models for a set of definite clauses P is still a Herbrand model of P. The intersection of all the Herbrand models of P is called the *Least Herbrand Model* of P and is represented with $LHM(P)$. Since LHM is a subset of each Herbrand model of a program, a clause satisfied by LHM is also satisfied by every Herbrand model of P. Hence, the least Herbrand model provides a way to find all the logical consequences of P: $P \models A$ iff $A \in LHM(P)$ [Rig98].

In addition to model theory, that provides means for deciding truth values of sentences, proof theory is also required to be considered, that deals with the generation of sentences (called *conclusions*) from other sentences (called *premises*). More specifically, proof theory considers the derivability of sentences by using some set of *inference rules*, that is, rules that decide what are the conclusions that can be obtained from a set of premises. Since proof theory can be applied to ground formulas, it can be employed to execute the coverage test of examples over logic programs. If a sentence A can be derived from a set of sentences P, we write $P \vdash A$. A proof, called *derivation*, is a sequence of sentences $s_1, \ldots, s_n$, such that each s_i is either in the premises or is derivable using the inference rules from the premises and $s_1, \ldots, s_{i-1}$.

Relation between logical entailment (or semantic entailment, $\models$) and logical derivation (or syntactic entailment, $\vdash$) is established by the following properties:

- ***soundness property***: a set of inference rules R is *sound* if all the sentences derivable from a set of premises P by means of R are logical consequences of P, that is, if $P \vdash s$ then $P \models s$;
- ***completeness property***: a set of inference rules R is *complete* if all the sentences logically following from a set of premises P are also derivable from P by means of R, that is, if $P \models s$ then $P \vdash s$.

The ideal case is to have a set of inference rules sound and complete with respect to logical entailment, where semantic consequences of a logic program can be derived in a syntactic way.

A proof procedure for clausal programs is *resolution*. It, starting from any two clauses, derives a new clause as their consequence. For example, the clauses $father(X, Y) \leftarrow male(X), parent(X, Y)$ and $male(john) \leftarrow$ resolve into the clause $father(john, Y) \leftarrow male(john), parent(john, Y)$.

Resolution is sound (every consequence is implied by its premises). However, it is not complete but only *refutation-complete*: if a set of clauses is inconsistent, resolution is able to derive the unsatisfiable empty clause.

3.3.1 Learning logical theories

This section focusses on the aspects concerning the learning of logical theories from examples, presenting the main issues that must be tackled in theories induction, where the basic goal is to find a set of logical clauses (or *theory*, $\mathcal{T}$) describing some target concepts starting from training (positive and negative) examples and a

given background knowledge (BK) expressing constraints and rules of the working domain. The set of training examples E is assumed to be composed by positive examples E^+ and negative examples E^- for the concepts to learn ($E = E^+ \cup E^-$).

Concept learning in ILP can be seen as a search problem in the hypothesis space that, in order to be scanned in a systematic way, requires the definition of a partial order over clauses. This partial order is provided by a generality relation imposed over clauses: generally speaking, a clause C_1 is *more general than* a clause C_2, represented with $C_1 \prec C_2$, if the set of examples covered by C_2 is a subset of those covered by C_1. The generality relation, however, is dependent from the particular setting where we are working. In practice, ILP systems implement the generality test by using a syntactic relation, called *θ-subsumption*, in place of entailment [Plo69].

Definition 3.1 *Clause C_1 θ-subsumes C_2 if there exists a substitution θ such that $C_1\theta \subseteq C_2$. We also say that C_1 is a generalization under θ-subsumption of C_2.*
Two clauses C_1 and C_2 are θ-subsumption equivalent, denoted with $C_1 \sim C_2$, if C_1 θ-subsumes C_2 and C_2 θ-subsumes C_1.

As a consequence, hypothesis space can be considered as a set of hierarchies, one for each concept to learn. In each hierarchy, the root represents the most general clause (with an empty body), each node represents a clause that can be induced and an edge connects two nodes if the clause corresponding to one of them can be induced by the other one. Such hierarchies can be explored in two alternative directions: in a top-down (or general to specific) fashion, starting from the root node and going towards the deeper levels by applying one or more refinement operators, or in a bottom-up fashion, starting from one of the more specific clauses (corresponding to a training example) and generalizing it until a stop criterion is satisfied or the root clause is reached.

However, the most important issue that must be faced when learning logical theories is the high computational cost, in term of both time and space required, due to the huge dimension of hypothesis space to be searched. To fight such complexity, ILP systems impose some constraints that aim at reducing the number of candidate hypotheses to be considered [Fla98]. Such constraints are known in literature as *declarative bias*. It consists of the *language bias*, determining the hypothesis space, and the *search bias* which restricts the search of the space of the possible hypotheses. Essentially, the main source of complexity in ILP derives from the variables in hypothesis clauses. In top-down systems, the branching factor of the specialization operator increases with the number of variables in the clauses. Here, arguments typing, that is, assigning a type to each predicate argument, can be useful to put a bound on the number of possible substitution that can be performed in clause specializations.

Other than typing, another way to limit the hypotheses to explore is to fix *mode declarations* of predicates, that is to describe possible input-output behaviour of a predicate definition. For instance, in a sorting program a mode declaration of the predicate $sort(+list, -list)$ means that the first argument must be instantiated with an already existing list, while the second one can be a uninstantiated variable.

Refinement operators can be also used as a language bias, since they can be restricted to generate only a subset of the language, or they can put a bound on the number of distinct variables that can occur in a clause.

According to the form in which examples are expressed and the coverage test that is adopted, two main settings have been proposed in literature: *learning from entailment* and *learning from interpretations*.

In *learning from entailment* setting, the training set is expressed as a set of ground facts, where each fact corresponds to an example, BK and hypothesis are definite programs and the coverage relation is defined as follows:

Definition 3.2 *Given a background knowledge BK, a hypothesis H and an example set E, H covers the example $e \in E$ w.r.t. BK if $BK \cup H \models e$.*

We say that a hypothesis H is *complete* if $\forall e^+ \in E^+ : BK \cup H \models e^+$ and is *consistent* if $\forall e^- \in E^- : BK \cup H \not\models e^-$, where E^+ denotes the positive examples in E and E^- the negative ones.

The framework of learning from entailment has also been called *normal* (or *explanatory*) *setting*.

The task of learning from entailment can be defined as follows:

Definition 3.3 *Given:*

- *a set $\mathcal{H}$ of possible programs (language bias)*
- *a set E^+ of positive examples (ground facts)*
- *a set E^- of negative examples (ground facts)*
- *a logic program BK (background knowledge)*

Find a logic program $H \in \mathcal{H}$ such that:

- *$\forall e^+ \in E^+ : BK \cup H \models e^+$ (completeness)*
- *$\forall e^- \in E^- : BK \cup H \not\models e^-$ (consistency)*

But, given two hypotheses, how can we state which of them is more general than the other one? In learning from entailment setting, given two hypotheses, H_1 and H_2, H_1 is more or equally general than H_2 w.r.t. the background knowledge BK, denoted with $H_1 \prec H_2$, iff $BK \cup \{H_1\} \models \{H_2\}$.

In *learning from interpretations* setting (also called *non-monotonic* setting) [BRJD99], each example is a Herbrand interpretation (i.e., a set of ground facts) and represents an independent observation relative to a particular situation in the world. The coverage test is defined as follows:

Definition 3.4 *Given a background knowledge BK, a hypothesis H and an example set E, H covers the example $e \in E$ w.r.t. BK if $LHM(BK \cup e) \models H$.*

The task of learning from interpretations can be defined as follows:

Definition 3.5 *Given:*

- *a set $\mathcal{P}$ of possible clausal theories (language bias)*
- *a set E^+ of positive examples (interpretations)*
- *a set E^- of negative examples (interpretations)*
- *a logic program BK (background knowledge)*

Find a clausal theory $P \in \mathcal{P}$ such that:

- $LHM(BK \cup e^+) \models H$ *(completeness)*
- $LHM(BK \cup e^-) \not\models H$ *(consistency)*

In learning from interpretations setting, given two hypotheses, H_1 and H_2, H_1 is more or equally general than H_2, denoted with $H_1 \prec H_2$, iff $H_2 \models H_1$. In fact, according to the definition of entailment, all the interpretations that are models for H_2 are also models for H_1. Hence, all the examples covered by H_2 will also be covered by H_1.

As a consequence, in learning from entailment setting, the induced hypotheses can always be used to replace the examples because BK and hypotheses entail the observed examples (as well as other unobserved examples). On the other hand, in the learning from interpretation setting the hypotheses consist of a set of properties holding for the example set, so there are no requirements nor guarantees concerning prediction [RL96]. Let us illustrate this concept with an example:

Example 3.1 *Suppose we have tweety, the canary, and oliver, the penguin. We know that both tweety and oliver are birds and that tweety flies while oliver does not. We want to find a characterization of flying animals. Formally, the available information can be represented as follows:*
$E^+ = \{flies(tweety)\}$
$E^- = \{flies(oliver)\}$
$BK = \{bird(tweety), bird(oliver)\}$
Suppose that the hypotheses to evaluate are:
$H_1 = flies(X) \leftarrow bird(X)$
$H_2 = bird(X) \leftarrow flies(X)$
In learning from entailment setting, H_1 is not a solution because $BK \cup H_1$ entails both E^+ and E^-, while H_2 cannot say anything about the examples.
In learning from interpretations setting, H_1 is still not a solution because there exists at least one model holding for $LHM(BK \cup E)$ that does not hold for H_1. In fact, by considering the substitution $\theta = \{X/oliver\}$, $body(H_1)\theta = bird(oliver) \subseteq LHM(BK \cup E^+)$ and $head(H_1)\theta = flies(oliver) \not\subseteq LHM(BK \cup E^+)$. Conversely, the clause H_2 is a solution because for each X that makes $flies(X)$ true, $bird(X)$ is also true. However, H_2 expresses a sufficient property of flying animals but cannot be used for prediction purposes.

3.3.2 Dealing with recursion

In the above section, the general aspects concerning the learning of logical theories have been presented. Nevertheless, real-world domains sometimes requires to deal with concept definitions that are inherently recursive. In some contexts, such as family relations, number characterization or document understanding [ABC+07], patterns occur repetitively in the same training observation, so the best way to capture a precise model is by means of recursive programs.

In this work, the recursive theory learning problem encompasses not only *strictly recursive definitions* (where the same predicate simultaneously occurs in the antecedent and the consequent of a clause), but also *mutual recursive definitions* (where two predicates are defined in terms of each other) or simply *dependent definitions* (where a target predicate occurs in the antecedent of at least one clause defining another target concept), since all these learning problems are equivalent.

For instance, concerning the number characterization domain, the only way to describe odd and even numbers is by relating each concept with the other one, that is, by means of the following (mutual) recursive definitions:

$odd(X) \leftarrow succ(Y, X), even(Y)$

$even(X) \leftarrow succ(Y, X), odd(Y)$

$even(X) \leftarrow zero(X)$

Analogously, to describe family relations we have to describe the *ancestor* concept in term of itself:

$ancestor(X, Y) \leftarrow ancestor(X, Z), parent(Z, Y)$

It has been proven that learning recursive definitions is equivalent to learning multiple concept definitions.

Moreover, most of times ILP learning systems are applied in domains where few or no information are available and the hope is to discover new knowledge by means of machine learning tools. The fact that one generally does not know in advance whether recursion is beneficial or not in a given application domain seems to justify the use of more general-purpose learning techniques that can induce both recursive and non-recursive theories.

However, dealing with recursion requires additional issues to be faced [Mal03]:

1. learning recursive definition (as well as multiple concept learning) is more difficult than learning single predicates, since the order in which concepts are learned affects the quality of induced theory

2. the generality order adopted in simple predicate learning, namely θ-subsumption, is no longer suited to deal with multiple predicate definitions

3. an additional property of multiple predicate learning, named *non-monotonicity property*, must be taken into account while learning

Firstly, whenever the learning task consists in the induction of more than one concept definition, an additional issue arises concerning the order in which the hypothesis spaces related to the target concepts must be explored. In fact, concept to learn might be someway related each other, so it is crucial that they are learned

from the independent ones towards the dependent ones. This corresponds to learn the concepts following the dependencies among them. For instance, in the family relations domain the concept definitions to learn should be:

$ancestor(X,Y) \leftarrow ancestor(X,Z), parent(Z,Y)$
$ancestor(X,Y) \leftarrow parent(X,Z)$
$father(X,Y) \leftarrow parent(X,Y), male(X)$
$grandfather(X,Y) \leftarrow father(X,Z), parent(Z,Y)$

It is clear that such predicate definitions can be learned only if the concepts are learned in the right order, because, for example, the learning of $grandfather$ needs that the concept $father$ has already been learned. This means that either the concept dependencies must be provided in advance or that a pre-processing step is required that has to discover the dependencies among concept to learn. A wrong hypothesis on predicate dependencies may affect or even prevent the learning results.

Secondly, the generality order typically used in ILP settings, namely θ-subsumption [Plo69], is not sufficient to guarantee the completess and consistency of learned definitions with respect to logical entailment [Got87, Mug92, ND96]. The main problem with the well-known θ-subsumption is that the elements of comparison are two clauses and no additional source of knowledge (e.g., a theory $\mathcal{T}$) is considered. Rather, we are only interested in those generality orders that compare two clauses relatively to a given theory $\mathcal{T}$. A lot of generality orders have been explored, such as the Buntine's *generalized subsumption* [Bun88] or the Plotkin's *relative generalization* [Plo69, Plo71], but the only one resulting to be adequate to deal with recursion has been proven to be the *generalized implication* [Mal03]:

Definition 3.6 *Let C and D be two definite clauses. C is more general than D under generalized implication, with respect to a theory $\mathcal{T}$, denoted as $C \leq_{\mathcal{T},\Rightarrow} D$, if a substitution θ exists such that $head(C)\theta = head(D)$ and $\mathcal{T} \models \forall(C\theta \Rightarrow D)$.*

Thirdly, an important problem in multiple predicate learning is the so called *non-monotonicity property*: whenever two individual clauses are consistent in the data, their conjunction need not to be consistent in the same data [DD97]. In other words, this property states that the consistency is not guaranteed to be preserved by adding new clauses to a theory $\mathcal{T}$.

Example 3.2 *For instance, the two clauses*
$C_1 = happy(X) \leftarrow loves(Y,X), woman(Y)$
$C_2 = woman(X) \leftarrow hag(X)$
are individually consistent with respect to:
$BK = \{loves(betty, george), loves(pamela, mark), hag(pamela)\}$
$E^+ = \{happy(george), woman(betty)\}$
$E^- = \{happy(mark)\}$
while the logical theory $\mathcal{T} = \{C_1, C_2\}$ is not because $BK \cup \mathcal{T} \models happy(mark)$.

As a consequence of the non-monotonicity property, clauses supplying predicates with multiple definitions should not be learned individually but, in principle, they should be generated all together.

In order to overcome these problems, De Raedt and Lavrač [RL96] have proposed the non-monotonic setting of ILP, where clauses can be investigated independently of each other, since their interactions are no longer important. However, this setting produces properties of examples instead of concept definitions from examples, as shown in Section 3.3.1.

3.4 ATRE: an example of recursive logical theory learner

The above mentioned aspects concerning the learning of logical theory and the additional issues arising when dealing with recursion have been faced in the ATRE system.

ATRE [Mal03] is a multiple-concept learning system that, given a set of concepts to be learned (or *target concepts*), a background knowledge on the working domain, a set of positive and negative examples for the target concepts and some preference criteria that guide the search in the hypothesis space, induces a (possible recursive) logical theory that describes all the positive training examples without covering any negative one. In addition, differently from most of available ILP learning systems, ATRE can deal with numerical attributes other than symbolic ones. In this case, a generalization of a numerical attribute describing the training examples is represented by the numerical interval that best discriminates the positive examples from the negative ones.

More formally, the learning task solved by ATRE can be defined as follows:

Definition 3.7 *Given:*

- *a set of concepts* $C_1, C_2, \ldots, C_r$ *to be learned*
- *a set of positive examples* E_i^+ *and negative examples* E_i^- *for each target concept* C_i
- *a background knowledge* BK *described in a language* $\mathcal{L}_{BK}$
- *a language of hypotheses* $\mathcal{L}_H$ *that defines the space of hypotheses* S_H
- *a set of user's preference criteria* PC

Find:
a (possibly recursive) logical theory $\mathcal{T} \in S_H$*, defining the concepts* $C_1, C_2, \ldots, C_r$*, such that* $\mathcal{T}$ *is complete and consistent with respect to the set of examples and satisfies the preference criteria* PC

The learning strategy
The high-level learning algorithm adopted by ATRE is called *separate-and-parallel-conquer*. It belongs to the family of *sequential covering* (or *separate-and-conquer*) algorithms [Mit97, Fur99] since it is based on the strategy of learning one clause at a time, removing the covered examples and iterating the process on the remaining

examples. Indeed, a recursive theory $\mathcal{T}$ is built step by step, starting from an empty theory $\mathcal{T}_0$, and adding a new clause at each step. In this way we get a sequence of theories

$$\mathcal{T}_0 = \emptyset, \mathcal{T}_1, \ldots, \mathcal{T}_i, \mathcal{T}_{i+1}, \ldots, \mathcal{T}_n = \mathcal{T}$$

such that $\mathcal{T}_{i+1} = \mathcal{T}_i \cup \{C\}$ for some clause C. If we denote by $LHM(\mathcal{T}_i)$ the Least Herbrand Model of a theory $\mathcal{T}_i$, the stepwise construction of theories entails that $LHM(\mathcal{T}_i) \subseteq LHM(\mathcal{T}_{i+1})$, for each $i \in \{0, 1, \ldots, n-1\}$, since the addition of a clause to a theory can only augment the LHM. Henceforth, we will assume that both positive and negative examples of predicates to be learned are represented as ground atoms with a $+$ or $-$ label. Therefore, examples may or may not be elements of the models $LHM(\mathcal{T}_i)$. Let $pos(LHM(\mathcal{T}_i))$ and $neg(LHM(\mathcal{T}_i))$ be the number of positive and negative examples in $LHM(\mathcal{T}_i)$, respectively. If we guarantee the following two conditions:

1. $pos(LHM(\mathcal{T}_i)) < pos(LHM(\mathcal{T}_{i+1})), \forall i \in \{0, 1, \ldots, n-1\}$, and
2. $neg(LHM(\mathcal{T}_i)) = 0, \forall i \in \{0, 1, \ldots, n\}$,

then we can state that, after a finite number of steps, a theory $\mathcal{T}$ complete and consistent w.r.t. the set of training examples is built.

In order to guarantee the first of the two conditions it is possible to proceed as follows. First, a positive example e^+ of a predicate p to be learned is selected, such that e^+ is not in $LHM(\mathcal{T}_i)$. The example e^+ is called *seed*. Then the space of definite clauses more general than e^+ is explored, looking for a clause C, if any, such that $neg(LHM(\mathcal{T}_i \cup \{C\})) = \emptyset$. In this way we guarantee that the second condition above holds as well. When found, C is added to $\mathcal{T}_i$ giving $\mathcal{T}_{i+1}$. If some positive examples are not included in $LHM(\mathcal{T}_{i+1})$ then a new seed is selected and the process is repeated.

The second condition is more difficult to guarantee because of the third issue of recursive theory induction problem presented above, namely, the non-monotonicity property. The approach followed in ATRE to remove inconsistency due to the addition of a clause to the theory consists of theory layering, that is, simple syntactic changes in the theory leading to new predicates addition. Details on the layering approach and on the computation method are reported in [Mal03]. The layering of a theory introduces a first variation of the classical separate-and-conquer strategy sketched above, since the addition of a locally consistent clause generated in the conquer stage is preceded by a global consistency check. In this way, ATRE overcomes the non-monotonicity property of ILP setting, guaranteeing that both partial and final logical theories are consistent with respect to all the training set of examples.

As concerns the generality order adopted to organize the search of hypothesis space, ATRE resorts to *generalized implication* which results to be the only one suited for recursive settings.

Concerning the first issue in the recursive theory induction, a solution to the problem of automated discovery of dependencies between target predicates $p_1, p_2, \ldots,$

p_r is based on the variant of the separate-and-conquer learning strategy that is adopted. Traditionally, this strategy is adopted by single predicate learning systems that generate clauses with the same predicate in the head at each step. In multiple/recursive predicate learning, clauses generated at each step may have different predicates in their heads. In addition, the body of the clause generated at the i-th step may include all target predicates $p_1, p_2, \ldots, p_r$ for which at least a clause has been added to the partially learned theory in previous steps. In this way, dependencies between target predicates can be generated.

Obviously, the order in which clauses of distinct predicate definitions have to be generated is not known in advance. This means that it is necessary to generate clauses with different predicates in the head and then to pick one of them at the end of each step of the separate-and-conquer strategy. Since the generation of a clause depends on the chosen seed, several seeds have to be chosen such that at least one seed per incomplete predicate definition is kept. Therefore, the search space is actually a forest of as many search-trees (called *specialization hierarchies*) as the number of chosen seeds. A directed arc from a node C to a node C' exists if C' is obtained from C by a single refinement step. Operatively, the (downward) refinement operator considered by ATRE consists in adding a new literal to a clause.

The forest can be processed in parallel by as many concurrent tasks as the number of search-trees (this motivates the name of *separate-and-parallel-conquer* for this search strategy). Each task traverses the specialization hierarchy top-down (or general-to-specific), but synchronizes traversal with the other tasks at each level. Initially, some clauses at depth one in the forest are examined concurrently. Each task is actually free to adopt its own search strategy, and to decide which clauses are worth to be tested according to a beam search adopted to limit the exponential growth of clauses to explore. If none of the tested clauses is consistent, clauses at depth two are considered. Search proceeds towards deeper and deeper levels of the specialization hierarchies until at least a user-defined number of consistent clauses is found. Task synchronization is performed after that all "relevant" clauses at the same depth have been examined. A supervisor task decides whether the search should carry on or not on the basis of the results returned by the concurrent tasks. When the search is stopped, the supervisor selects the "best" consistent clause according to the user's preference criteria. This strategy has the advantage that simpler consistent clauses are found first, independently of the predicates to be learned. Moreover, the synchronization allows tasks to save much computational effort when the distribution of consistent clauses in the levels of the different search-trees is uneven.

Since the hypothesis space to explore is too wide, both the width of the beam search (that is, the number of clauses to be specialized) and selection of the best clause to be added in the learned theory rely on various elementary, easy-to-measure criteria specifying desirable properties for establishing which clause is better than another one. This criteria, each of them measuring a certain aspect of the generated clauses, are assembled together into one general criterion, called *Lexicographical Evaluation Functional with tolerances* (*LEF*). In other words, the LEF consists of an ordered sequence of elementary criteria along with tolerances that control to what

extent different solutions are considered equivalent. First, all clauses are evaluated according to the first elementary criterion. Those that score best or within the given tolerance range from the best are retained. Those retained are then evaluated according to the next elementary criterion, and so on, until either a single clause remains or the list of elementary criteria in the LEF is exhausted. In the latter case, all clauses that remain are judged equal and the algorithms picks one arbitrarily.

An example of the search strategy adopted by ATRE in the parallel exploration of the hypotheses for the odd and even concept definitions is showed in Fig. 3.2. The Figure represents two learning step: in the former the hierarchies are explored untill a sufficient number of consistent clauses (defined by a user defined parameter) has been generated; after that, the best clause according to the preference criteria is selected as learned clause ($even(X) \leftarrow zero(X)$, in the example), example descriptions are enriched with the literals, if any, resulting from the application of the learned clause on training dataset (*saturation* step), and another learning step is performed starting from the root clauses. The second learning step is performed in the same way, yielding the clause $odd(X) \leftarrow succ(Y, X), even(Y)$.

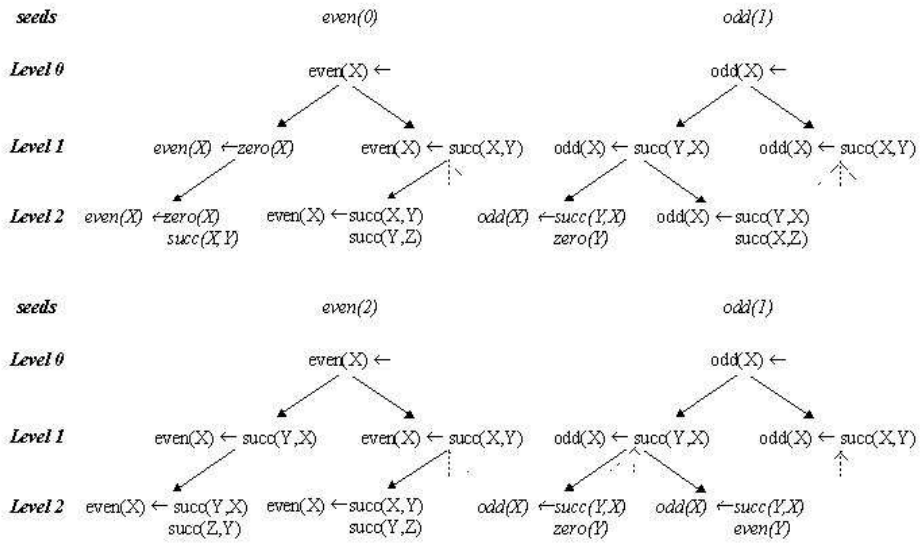


FIGURE 3.2: ATRE: the parallel exploration of the specialization hierarchies for odd and even concepts

Representation formalism

Concerning the representation formalism adopted by ATRE, the basic component of representation language is the *literal* or simple term, i.e. a function symbol where arguments are either constants or variables (it can not be a compound term). Two kind of literals are recognized by the system:

- $f(t_1, \ldots, t_n) < op > Value$
- $g(s_1, \ldots, s_n)$ *in Range*

where f and g are function symbols called *descriptors*, each t_i and s_i is either a

constant or a variable, *op* is a relational symbol (=, >, <, >=, <=), $Value$ is a value in the codomain of f and *Range* is either a numeric interval or a finite set of values of g.
Some examples are:

$color(X1) = red$
$height(X1)\ in\ [1.1..1.2]$
$ontop(X, Y) = true$

This example shows the lack of predicates in the ATRE representation language: each predicate p is associated with a function symbol f_p. Hence, first order literals $p(X, Y)$ and $\neg p(X, Y)$ are represented with $f_p(X, Y) = true$ and $f_p(X, Y) = false$, respectively.

Data representation

Training examples (or *observations*) recognized by ATRE are described according to an object-centered representation. Each *object* (modelling an observation) considers a complex system like a single entity represented with a multiple-head ground clause (a ground clause with a conjunction of simple literals in the head side). For instance, the following description is an object:

$type(blk1) = lintel \wedge type(blk2) = column \leftarrow pos(blk1) = hor, pos(blk2) = ver,$
$ontop(bk1, blk2)$

that is semantically equivalent to the following definite program:

$type(blk1) = lintel \leftarrow pos(blk1) = hor, pos(blk2) = ver, ontop(bk1, blk2)$
$type(blk2) = column \leftarrow pos(blk1) = hor, pos(blk2) = ver, ontop(bk1, blk2)$

This example clearly shows that the main advantages carried out by this representation consist of an improved comprehensibility and efficiency. Comprehensibility improves because each multiple-head clause provides a compact description of the properties to be predicted in a structured object, while efficiency improves because the properties of an object are represented once, saving both time and space resources. It is worth to be remarked that only the equality operator, =, is allowed to be present in the object: since the object represent a direct observation in the real world domain, descriptors can only get precise values of their own domain. Moreover, objects are represented in form of the literal $object(Id, Head, Body)$, where Id is an unambiguous object identifier while $Head$ and $Body$ are the literal lists representing the head and body of clause, respectively.

Background knowledge representation

The representation language concerning the BK is composed by connected and range-restricted definite clauses; these clauses can not refer to concepts to be learned in their body. BK is used to enrich the description of the training examples. Each object body is matched against the BK clauses with the aim of finding all the direct consequences. These consequences, in form of new literals, are added to the object body. This simple transformation process, called *saturation*, whose goal is to make explicit the information embedded in BK, is repeated until no further literals are added by the matching process.

Saturation can involve a change in the hypothesis language, in the sense that

the learned clauses can also be expressed by means of additional function symbols that are not directly used to describe the training examples. For this reason, ATRE can be reasonably classified like an ILP system that fulfil a kind of constructive induction.

As concerns the operators used in BK clauses, only equality operator is allowed to be used in the clauses head while no restriction is fixed for the body of clauses. Each BK clause is represented by means of the predicate $bk(No_{bk}, Head, Body)$, where No_{bk} is the BK clause identifier and $Head$ and $Body$ are the literal lists representing the head and body of BK clause, respectively.

Hypotheses representation

The hypotheses are represented by means of connected definite and range-restricted clauses. They can not contain constant terms and, as concerns the involved operators, only the equality operator can appear in the head of the clauses while both equality and inclusion (in) operators can appear in the body of clauses.

During the learning process a change in the hypothesis language is possible; the reason is that ATRE performs the saturation of training objects after each clause is learned in order to make the (mutually) recursive concepts learning possible.

The input file

To perform learning, ATRE needs an input file describing both the training data and the learning parameters. This is a text file (whose extension is '.dat'), containing prolog-like statements (each statement must be followed by a full stop), one for each declared parameter or training object. In the following, a list of available statements is presented (notice that in ATRE formalism the assignment operator is represented by '=.'):

- $concept\_list(list\_of\_concepts)$: this statement (that is mandatory) defines the list of concepts to be learned; concepts have to be expressed in the form $literal = .value$.

- $descriptor(literal(arg_1, \ldots, arg_N) = .value, lit\_type, weight)$: these statements define the literals the system is allowed to use in the learning process. If a literal definition is missing in the training file, no hypothesis containing such a literal can be formulated. arg_i and $value$ are place cards of arguments and literal value. lit_type specifies the literal type; it can be either $nominal$, $boolean$ or $real$. In case of real descriptor, the range in which the descriptor can be considered valid must be specified. $weight$ is a measure affecting the inclusion of the descriptor in the learned theory: for instance, if LEF criterion suggest to prefer lower cost clauses, higher values of this parameter discourage the learner from inserting the literal into the learned clauses.

- $best\_lef(selection\_criteria\_list)$: specifies the criteria according to which the learner must choose the best clause to add in the logical theory.
  $selection\_criteria\_list$ is the list of simple criteria for the selection. Each criterion must be expressed in the form $[criterion, min\_max, tolerance]$, where

min_max can be *min* or *max* (if the learner has to select the clauses that minimize or maximize the criterion, respectively), *tolerance* is a tolerance value (a real value in the range from 0 to 1) that can be used during the clause selection, and *criterion* can be one of the following:

- *poscov*: number of positive examples covered by the clause;
- *negcov*: number of negative examples covered by the clause;
- *numlet*: number of literals in the body of the clause;
- *range*: number of variables in the head of the clause, but not in the body;
- *domain*: number of variables in the body of the clause, but not in the head;
- *posiniz*: number of initial positive examples covered by the clause;
- *perneg*: rate of covered negative examples out of the overall training examples;
- *cost*: cost of a clause, calculated as the sum of weights of each literal in the clause;

Order in which the preference criteria are expressed is significant. If not defined, the default assertion for this statement is: $best\_lef([[poscov, max, 0.0]])$.

- $ps\_lef(selection\_criteria\_list)$: specifies the criteria according to which the learner must choose the set of clauses to be specialized. The syntax and semantics of criteria definition is the same as $best\_lef$ specification. The default assertion for this statement is: $ps\_lef([[negcov, min, 0.0], [poscov, max, 0.0]])$.

- $maxstar(value)$: specifies the maximum number of clauses to specialize in each specialization step; the higher is this number, the wider is the beam considered during the hypotheses space search. If not found in training file, the default value of this parameter is 5.

- $consistent(value)$: specifies the minimum number of clauses to be generated before the learner stops the hypotheses search and chooses the best one to add in the learned theory. The default value is 1.

- $max\_ps(value)$: indicates the maximum number of literals that can be added in a clause body and, hence, the maximum depth level in the specialization hierarchies the search can reach. The default value is set to 15.

- $verbosity(Objs, Exs, Clause\_selection, Generated\_clauses, Cons\_RR\_clauses)$: sets the information to trace into the report file. A report file can contain information about the training objects ($Objs$), training examples (Exs), selection process of clauses performed in each learning step ($Clause\_selection$), each clause generated by the learner during the hypotheses exploration ($Generated\_clauses$), consistent and range-restricted clauses found by the learner ($Cons\_RR\_clauses$).

- $bk(BK\_no, Head, Body)$: used to define each of clauses composing the background knowledge. BK_no represents a number identifying the clause, $Head$ is the list containing the literal in the head of BK clause (since we are dealing with definite clauses, this list must contain only one literal), and $Body$ is the list of literals in the body of clause we are specifying.

- $object(Id, Head, Body)$: describes a training object modelling each observation. Id represent the object identifier, $Head$ is the list of positive and negative examples in the object and $Body$ is the list of literals describing the positive and negative observed examples. Each training file must contain at least one object. The object definitions are the last statements in the training data file.

Optionally, we can either suggest or force ATRE to insert some literals into the learned clauses concerning one or more target concept. This is done by means of the following three statements:

- $starting\_number\_of\_literals(Concept, Lits)$: defines the minimum number $Lits$ of literals in the body of learned clauses concerning the concept $Concept$ to be learned. In other words, it fixes the number of literals in the clauses representing the roots of specialization hierarchies of concept $Concept$.

- $starting\_literal(Concept, Lits\_list)$: specifies the list of literals that can be found in the body of clauses representing the specialization hierarchy roots of $Concept$. For instance, if $Lits\_list$ is equal to $[Lit_1, Lit_2, \ldots, Lit_N]$, we are asking ATRE to consider clauses defining the concept $Concept$ with at least $Lits$ literals (where $Lits$ is specified in $starting\_number\_of\_literals$ statement); such literals can be Lit_1 or Lit_2 or $\ldots$ or Lit_N. In addition, for each literal in the list, the mode declaration (*old* or *new*) of each argument must be specified.

- $starting\_clause(Concept, Lits\_list)$: specifies the list of literals that are requested to be found in the body of clauses representing the specialization hierarchy roots of $Concept$. Differently from the $starting\_literal$ statement, the presence of all literals defined in $Lits\_list$ is mandatory. Thus, if $Lits\_list$ is equal to $[Lit_1, Lit_2, \ldots, Lit_N]$, the statement states that for the concept $Concept$ the root clauses must contain the literal Lit_1 and Lit_2 and $\ldots$ and Lit_N.

Finally, a well-know limit of ILP systems is their inefficiency, both in term of time and space required. ATRE particularly suffers from efficiency problems, since its ability to deal with recursion and concept dependencies further increases the computational complexity of the learning task. One source of inefficiency is that the same portion of hypothesis space is examined different times during learning, since at every learning step the search starts from scratch. In order to restrain such drawback, some caching techniques have been adopted in the system that aim at storing information about the portion of hypothesis space already explored and employ such information in the following hypothesis search [BVM04, VBM04a, VBM04b].

3.4.1 Learning from positive examples with ATRE

The ILP learning system ATRE, presented in the previous section, has been developed to perform a supervised learning task where both positive and negative examples are provided as training instances of the target concepts. It is well-known in machine learning that positive examples are used by the learner to prevent from learning too specific theories (*overfitting*) while negative examples are used to avoid to generate too general theories.

The question that arises is: can we employ ATRE to learn from only positive examples?

It is clear that, in general, this is not possible because the absence of negative examples would lead all generated hypothesis to be consistent and, hence, the search for the best clause to choose at each learning step would always return the simplest clause (with no literals in the body).

Nevertheless, ATRE can be able to learn *models* (conceived as logical theories) by only positive examples but this requires a specific setting of learning parameters described in the previous section.

First of all, we must properly set the LEF criteria in order to prefer the most specific clauses instead of the simpler ones. To this aim, a possible $best\_lef$ and ps_lef specification could be:

$$best\_lef([[poscov, max, 0.0], [numlet, max, 0.0], [cost, min, 0.0]])$$
$$ps\_lef([[poscov, max, 0.0], [numlet, max, 0.0], [cost, min, 0.0]])$$

In this way, we are forcing the system to choose the clauses that maximize the covered examples ($[poscov, max, 0.0]$), that have the highest number of literals in the body as possible and, thus, to select the most specific clauses ($[numlet, max, 0.0]$), and that have the minimum cost in term of literal weights ($[cost, min, 0.0]$) both in the specialization steps and in the choice of best clause to add in the theory.

In this simple example, the same criteria have been defined for both the LEF criteria, but user can tune these parameters to find the solution that best fit his/her needs.

Other than properly fixing the LEF criteria, we must induce the system to reach the deeper level of specialization hierarchy, taking into account that ATRE performs a top-down exploration of hypothesis space. To this aim, we have to set high values for the *consistent* and max_ps parameters and a low value for the *maxstar* parameter. In fact, the *consistent* parameter should be high in order to stop the search for clauses only when a remarkable number of clauses have been generated, while the max_ps parameter should be high enough to guarantee an acceptable depth of the last level explored. Finally, the *maxstar* parameter should tend to be as low as possible in order to make as thin as possible the exploration beam.

In the next chapter, we will see how ATRE can be profitably used to induce models over data that are used by the clustering method discussed in this thesis.

3.5 Conceptual clustering

As mentioned at the beginning of this chapter, most studies concerning relational data mining have been focussed on a particular clustering approach, called *conceptual clustering*.

Generally speaking, clustering is a form of unsupervised learning from observation, that is, learning without a teacher, where examples are provided without specifying their class membership and the goal is to structure them into meaningful categories. Past work on this problem was mostly done under numerical taxonomy and cluster analysis. These methods are based on the application of a mathematical measure of similarity between objects, defined over a finite, a priori given set of object attributes. Class of objects are taken as collections of objects with high intraclass and low interclass similarity. These methods, since mostly rely on propositional representation, often suffer from limitations coming from this form of representation, that is they are inadequate for dealing with structured objects and cannot take into consideration any background knowledge.

In contrast to traditional clustering approaches, *conceptual clustering* generates classes, corresponding to clusters of objects, by first generating conceptual descriptions of the classes and then classifying the objects according to these descriptions. Here, the basic idea is that objects should be arranged into classes representing simple concepts rather than classes defined solely by a predefined measure of similarity among their members.

In [SM86] Stepp and Michalski propose to extend conceptual clustering as follows:

- objects and classes are described by structural descriptions expressed in a typed predicate calculus, called *Annotated Predicate Calculus* (*APC*)
- background knowledge is used to derive high-level descriptive concepts from the low-level concepts provided in the example descriptions
- the system is supplied with a general goal of the classification, in order to obtain clusters that better satisfy the user expectations

In particular, the BK consists of a network of inference rules and heuristics for deriving new descriptions other than a network of classification goals, called *Goal Dependency Network* (*GDN*) that is used for guiding the search for relevant descriptions. Lexicographical Evaluation Functional with tolerances (LEF) is used to select the classification scheme that is the most preferred according to the given goal among the available schemes.

Consequently, Stepp and Michalski propose two different clustering methods:

- *Repeated Discrimination* (RD)
- *Classifying Attribute* (CA)

The *RD* method reduces the problem of building clusters into a sequence of problems of determining discriminant descriptions of objects with given class labels. In particular, in the method two main steps are iteratively applied: in the first step

the learning system INDUCE/2 is used to find class descriptions from examples by exploring descriptions that discriminates examples belonging to a class from the other ones (specifically, examples belonging to class which description must be found are considered as positive examples, while the other ones are considered as negative examples); the second step applies CLUSTER/2 to descriptions generated by INDUCE/2 in order to form optimized classifications.

CA method attempts to find one or more classifying attributes whose value sets can be split into ranges that define individual classes. It works by alternating two steps: GENERATE process, that generates new attributes by applying the BK to the available attributes, and SEARCH process, that searches for classifying attributes and their value set partitions that best discriminates classes on the basis of LEF criterion.

Another conceptual clustering method is *KBG* [Bis92] that creates a similarity matrix representing the observed examples. In each step, the method uses the similarity matrix to detect the most similar examples, that are generalized and arranged in the same cluster. After that, the similarity matrix is updated by replacing the examples with their generalization and another step is performed until all the provided examples are grouped in only one class.

Finally, one of the most-known conceptual clustering in literature is *COBWEB* [Fis87], an incremental clustering method that arranges clusters into a classification tree. The system carries out a hill-climbing search through a space of hierarchical classification schemes using operators that enable bidirectional travel through this space. Such search is guided by a heuristic measure called *category utility*.

COBWEB incrementally incorporates objects into a classification tree, where each node is a probabilistic concept that represents an object class. The incorporation of an object is a process of classifying the object by descending the tree along an appropriate path, updating counts along the way, and performing one of the following operation at each level:

- *placing an object into an existing class*: the category that best hosts a given object is detected by tentatively placing the object in each category and assigning it to the category that yields the best partition according to the category utility.
- *creating a new class*: the quality of the partition resulting from placing the object in the best category is compared to the partition resulting from creating a new singleton class containing the object.
- *combining two classes into a single class*: merging two nodes involves creating a new node and summing the attribute-value counts of the nodes being merged. The two original nodes are made children of the newly created node. Only the two best categories are considered for merging.
- *dividing a class into several classes*: Splitting is the opposite of the previous operation and is considered only for children of the best category.

Empirical evaluations indicate that COBWEB is a cost effective means of learning classification trees, due to its incremental way of work. However, one limiting aspect of this method is the object description language adopted: attribute-value formalism prevent it to deal with structured object descriptions. In addition, it cannot handle numeric attributes and, hence, cannot fully be applied to complex real-world domains.

3.6 Conclusions

In this chapter, the main limitations of classical DM approaches have been investigated. In particular, since they adopted the propositional formalism for data representation, they often result to be inadequate to be applied to real-world domains, where observations are generally structured, that is, composed by a set of heterogeneous sub-elements and relations among them. To overcome such limitation, MRDM methodology has been developed that overcomes *single-table assumption* characterizing the propositional approaches and is able to discover patterns over multiple tables in a relational database.

Other than some well-known methods belonging to this approach, two methodologies have been outlined for moving from propositional towards the relational formalism. In particular, the problem to solve is to fill the gap between the propositional formalism in which most of developed methods in literature are able to work and the relational form in which usually data are available. The former approach aims at upgrading propositional learners to first order logic, while the latter approach performs a propositionalization of relational data.

In order to deal with structured data, MRDM typically resorts to Inductive Logic Programming (ILP). Hence, the basic definitions and concepts characterizing first-order logic and ILP have been introduced. A particular section has been devoted to present the issues related to logical theory induction problem and paying particular attention in the additional problems affecting the induction of recursive or multiple-predicate theories.

As an example of ILP (recursive) learning system, the ATRE system has been presented, showing how it works, what are its main peculiarities and learning parameters and how it can be configured to learn from only positive data.

In the next chapter, we will see how ATRE can be profitably used to induce models over data that are used by the clustering method discussed in this thesis.

Finally, the last section of this chapter has been devoted to present the conceptual clustering approach, where most studies of MRDM have been concentrated.

As a final remark, we can state that MRDM methodology, thanks to ILP formalism and its inference rules, has been proven to be able to capture patterns over structured data and, hence, is able to detect the basic logical morphology underlying the available observations. In this sense, it fills the gap of spatial clustering approaches concerning the form of data they can deal with. As a consequence, clustering results could be improved by combining the spatial and multi-relational approaches, that is the basic idea of CORSO, the clustering method presented in

the next chapter.

Chapter 4

Clustering related structured objects

Up to now, we have discussed the problem of grouping available data according to some similarity criteria with the aim of placing together similar objects and separating different ones. This problem corresponds, after all, to the need for arranging the huge amount of gathered data into a format more understandable and analysable for humans. We have seen that, according to the underlying philosophy and, in turn, the similarity concept considered, different approaches (and methods belonging to them) exist in literature that face this problem. Each of the presented approaches is characterized by strengths and drawbacks: in particular, we have analysed the spatial clustering approach that groups objects according to their spatial features (and relations deriving from them) but cannot deal with structured objects because of the single-table assumption. On the other hand, we have presented the multi-relational clustering approach, that overcomes the single-table assumption by resorting to ILP, but works in frameworks where each object is considered to be completely independent from the other ones and, hence, cannot capture in the mined patterns the continuity of socio-economic or geographic phenomena over spatial regions.

In this chapter, we present *CORSO* (*Clustering Of Related Structured Objects*), a spatial clustering method that combines the spatial and multi-relational approach in order to take advantage from the strengths of both of them and be able to group together structured (or multi-relational) spatial objects into meaningful classes. Its main characteristic is the construction of clusters where objects are arranged according to both their spatial relations and their structural resemblances. The output is not only the list of resulting clusters with the membership of each input object into a cluster, but also the model associated with each obtained cluster, that is, the logical theory (expressed in a first-order logic formalism) describing the common substructure of objects belonging it. In this way, CORSO provides users with the reason according to which each cluster has been created other than the arrangement of spatial objects into groups.

4.1 Background and motivations

The motivation that has given rise to CORSO is to build a spatial clustering approach that, differently from the available ones, is able to consider spatial entities as structured objects by overcoming the single-table assumption that characterizes the spatial clustering methods. In this way, it is possible to group objects not only on the basis of spatial features and relations (distance or connectivity, for instance), but also according to their structural similarity, leading to a more meaningful set of obtained clusters, where objects are grouped together not only because they are sufficiently close each others, but also because they are sufficiently similar.

In spatial framework, gathered data are usually associated with areas (expressed as either irregular partitions of the available space or regular grid) rather than points in the space.

Areal data can be represented as point data by identifying each area with its centroid [Vis83], but this is restrictive when observations for an area are descriptive of one or more (spatial) primary units, possibly of different types, collected within the same area boundary. In this case, data includes both attributes that relate to primary units or areas and attributes that refer to relations between primary units (e.g., contact frequencies between households) and between areal units (e.g., migration rates). Moreover, spatial-referencing poses a further degree of complexity due to the fact that the geometrical representation (point, line or polygon) and the relative positioning of primary units or areal units implicitly define spatial features (properties and relations) of different nature, that is, geometrical (e.g. area, distance), directional (e.g. north, south) and topological (e.g. crosses, on top) features. This relational information may be responsible for the spatial variation among areal units and it is extremely useful in descriptive modeling of different distributions holding for spatial subsets of data. An extra consequence is that observations across space cannot be considered independent due to the spatial continuity of events occurring in the space. Continuity of events over neighbor areas is a consequence of social patterns and environmental constraints that deal with space in terms of regions and allow to identify a mosaic of nearly homogeneous areas in which each patch of the mosaic is demarcated from its neighbors in terms of attributes levels. For instance, the spatial continuity of an environmental phenomenon such as air pollution may depend on the geographical arrangements of pollution sources. As a model for this spatial continuity, the regional concept encourages the analyst to exploit spatial correlation following from the first Law of Geography [Tob79], according to which "*everything is related to everything else, but near things are more related than distant things*". This means that primary units forming areal units of analysis will tend to be essentially identical members of same populations in nearby locations. In this spatial framework, relations among areal units of analysis are expressed in form of relational constraints that represent a *discrete spatial structure* arising in spatial data, while relations among primary units within an area model the spatial structure of each single areal unit of analysis.

The spatial structure above mentioned is said to be *discrete* to emphasize the fact

that most of spatial phenomena are continuous (air pollution, for instance) but we are interested in the discretization of such phenomena in order to generate models of them. Data structures employed to represent discrete phenomena are *tessellation* and *vector*. The former partitions the space into a number of cells each of which is associated with a value of a given attribute. No variation is assumed within a cell and values correspond to some aggregate function (e.g., average) computed on original values in the cell. A grid of square cells is a special tessellation model called *raster*. This model is simple but the geometry of a spatial object is imprecise and requires large storage capabilities. In the *vector* model the geometry is represented by a vector of coordinates. This is a concise and precise representation but involved data structures are complex and the computation of spatial operations, such as intersection, is computationally demanding.

We propose to represent the discrete spatial structure as a graph, where nodes are associated with relational descriptions of areal units to be clustered, while links express relational constraints which typically reflect spatial relations such as adjacency. In this way, discontinuity in the graph represents some obstacles in the space.

Considering the clustering strategy adopted and the formalism in which data and models are expressed, CORSO takes advantage from the contribution of different approaches, as illustrated in Figure 4.1.

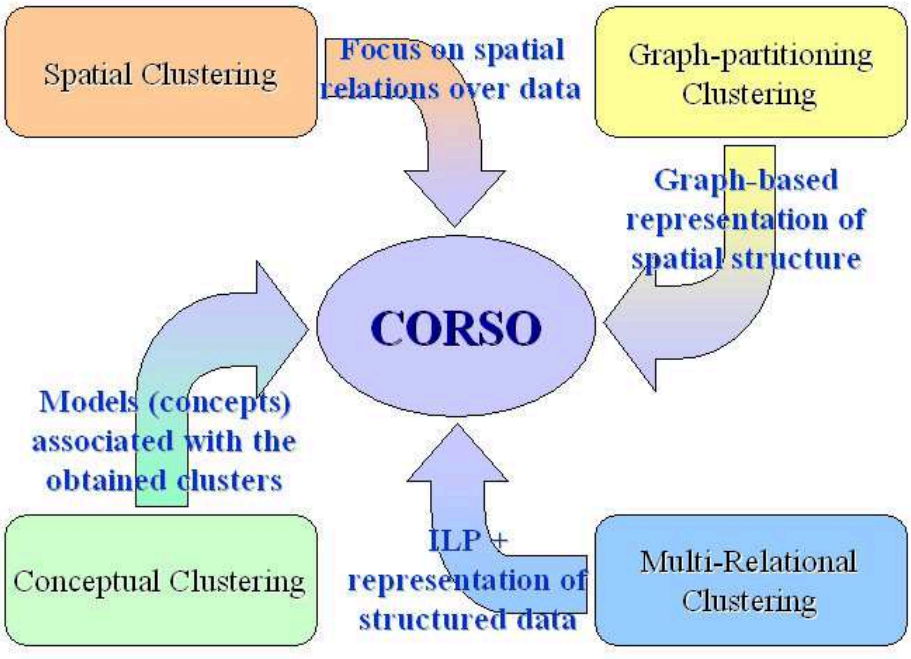


FIGURE 4.1: CORSO: clustering approaches contributions

From the spatial clustering approach, CORSO takes the basic requirement for the objects aggregation: the existence of spatial relations among the considered objects. As the main goal is to group together spatial objects, the first requirement

of a cluster is that each object belonging to it must be related to some other object in the same cluster. As a consequence, two objects not linked by any of the defined spatial relations cannot belong to the same cluster.

From the graph-based clustering approach, the method gets the graph-based representation as the structure adopted to easily and effectively represent spatial objects and their relations (discrete spatial structure). Consequently, the problem of detecting clusters of objects can be reformulated as a graph-partitioning problem, where the goal is to find the best partitioning of the graph such that each partition includes objects resulting to be connected and similar according to a given measure. However, differently from the graph-partitioning approaches where partitions are calculated by only taking into account links among nodes, our purpose is consider structural similarity of objects as well.

The contribution of conceptual clustering approach is that each detected cluster in CORSO is associated with an implicit definition (or model), expressed in first-order logic formalism, representing the objects belonging to it. Such models are used by the method to evaluate the homogeneity of the set of objects candidate to become a cluster (see Section 4.2.2). However, they are also provided as the output of the method since they can be considered as an intensional description of the clusters (in addition to the extensional description consisting of the list of objects belonging to the clusters).

Finally, the methods takes from the MRDM approach the capability of dealing with relational (or structured) objects and using the ILP framework as the basic mechanism to reason about them in order to induce their models (or generalizations).

4.2 The method CORSO

In a quite general formulation, the problem of clustering structured objects (e.g., complex areal units), which are related by links representing persistent relations between objects (e.g., spatial correlation), can be defined as follows:
Given:

- a set of structured objects O
- a background knowledge BK and
- a binary relation R expressing links among objects in O;

Find
a set of homogeneous clusters $\mathbf{C} \subseteq \wp(O)$ that is coherent with R.

Each structured object $o_i \in O$ can be described by means of a conjunctive ground formula (conjunction of ground selectors) in a first-order formalism (see Example 1), while background knowledge BK is expressed with first-order clauses that support some qualitative reasoning on O (see Example 2). In both cases, each basic component (i.e., *selector*) is a relational statement in the form $f(t_1, \ldots, t_n) =$

v, where f is a function symbol or *descriptor*, t_i are terms (constant or variables) and v is a value taken from the categorical or numerical range of f.

Structured objects are related by R that is a binary relation $R \subseteq O \times O$ imposing a discrete structure on O. In spatial domains, this relation may be either purely spatial, such as topological relations (e.g. adjacency of regions), distance relations (e.g. two regions are within a given distance), and directional relations (e.g. a region is on south of an other region), or hybrid, which mixes both spatial and non spatial properties (e.g. two regions are connected by a road).

Example 1. Let us consider data consisting of observations for a site (e.g., areal units) descriptive of one or more (spatial) primary units, possibly of different type, collected within the same site boundary. The areal units are the (structured) objects to be clustered, while the discrete data structure is naturally imposed by the spatial adjacency relation among areal units. Areal units are described in terms of both spatial and aspatial properties, such as:

$arealunit(apulia) \leftarrow contain(apulia, bari), is\_a(bari) = town,$
$inhabitants(bari) = 342129, contain(apulia, taranto), is\_a(taranto) = town,$
$distance(bari, taranto) = 98, \ldots$
$arealunit(basilicata) \leftarrow contain(apulia, potenza), is\_a(potenza) = town,$
$inhabitants(potenza) = 68141, contain(apulia, matera), is\_a(matera) = town,$
$\ldots$

In this case $adjacent(apulia, basilicata)$ and $adjacent(basilicata, apulia)$ are two instances of the relation R (adjacency relation).

Example 2. Background knowledge is a source of domain independent knowledge. For example the definite clause:

$$accessibility(X, Y) \leftarrow town(X) = XName, town(Y) = YName,$$
$$cross(X, Z) = true, cross(Y, Z) = true, road(Z) = ZName.$$

expresses accessibility of a town from another town by means of one road.

The relation R can be described by the graph $G = (N_O, A_R)$ where N_O is the set of nodes n_i representing each structured object o_i and A_R is the set of arcs $a_{i,j}$ describing links between each pair of nodes $\langle n_i, n_j \rangle$ according to the discrete structure imposed by R. This means that there is an arc from n_i to n_j only if $R(o_i, o_j)$ holds.

Let $N_R(n_i)$ be the *R-neighborhood* of a node n_i such that $N_R(n_i) = \{n_j |$ there is an arc linking n_i to n_j in $G\}$. Then, a node n_j is *R-reachable* from n_i if $n_j \in N_R(n_i)$, or $\exists n_h \in N_R(n_i)$ such that n_j is R-reachable from n_h.

According to this graph-based formalization, a clustering $\mathbf{C} \subseteq \wp(O)$ is *coherent* with the discrete structure imposed by R (or, shortly, coherent with R) when two objects o_1 and o_2 can belong to the same cluster C_i only if a linking path exists from o_1 to o_2 (or vice-versa) according to R.

Moreover, the cluster C is *homogeneous* when it groups structured objects of O sharing a similar relational description according to some similarity criterion.

CORSO integrates a neighborhood-based graph partitioning to obtain clusters which are coherent with the discrete structure defined by R and resorts to a multi-relational approach to evaluate similarity among structured objects and form homogeneous clusters. This faces with the spatial issue of modelling spatial continuity of a phenomenon over the space.

The top-level description of the method is presented in Algorithm 4.1.

Algorithm 4.1 Top-level description of CORSO algorithm.

```
1: function CORSO(O, BK, R, h − threshold) → CList;
2: CList ← ∅; O_BK ←saturate(O,BK); C ← newCluster( );
3: for each seed ∈ O_BK do
4:     if seed is UNCLASSIFIED then
5:         N_seed ← neighborhood(seed,O_BK,R);
6:         for each o ∈ N_seed do
7:             if o is assigned to a cluster different from C then
8:                 N_seed = N_seed/o;
9:             end if
10:        end for
11:        T_seed ← neighborhoodModel(N_seed);
12:        if homogeneity(N_seed, T_seed) ≥ h − threshold then
13:            C.add(seed); seedList ← ∅;
14:            for each o ∈ N_seed do
15:                C.add(o); seedList.add(o);
16:            end for
17:            ⟨C, T_C⟩ ←expandCluster(C,seedList,O_BK,R,T_seed,h − threshold);
18:            CLabel=clusterLabel(T_C); CList.add(⟨C, CLabel⟩); C ← newClus-
   ter( );
19:        else
20:            seed ← NOISE;
21:        end if
22:    end if
23: end for
24: return CList;
```

CORSO embeds a saturation step (function *saturate*) to make explicit information that is implicit in data according to the given BK. New information (in form of literals) is added to object descriptions by repeatedly applying BK rules to available descriptions of data until no additional literals can be derived from the application of the BK.

The key idea is to exploit the R-neighborhood construction and build clusters coherent with R-discrete structure by merging partially overlapping homogeneous neighborhood units. Cluster construction starts with an empty cluster ($C \leftarrow newCluster()$) and chooses an arbitrary node, called *seed*, from G.
The R-neighborhood N_{seed} of the node *seed* is then built according to G discrete structure (function *neighborhood*) and the first-order theory T_{seed} is associated to

Algorithm 4.2 Expand current cluster by merging homogeneous neighborhood.

```
   function expandCluster(C, seedList, O_BK, R, T_C, h − threshold) → ⟨C, T_C⟩;
 2: while (seedList is not empty) do
      seed ← seedList.first(); N_seed ← neighborhood(seed,O_BK,R);
 4:   for each o ∈ N_seed do
        if o is assigned to a cluster different from C then
 6:       N_seed = N_seed/o;
        end if
 8:   end for
      T_seed ← neighborhoodModel(N_seed);
10:   if homogeneity(N_seed, {T_C, T_seed}) ≥ h − threshold then
        for each o ∈ N_seed do
12:       C.add(o); seedList.add(o);
        end for
14:     seedList.remove(seed); T_C ← T_C ∪ T_seed;
      end if
16: end while
   return ⟨C, T_C⟩;
```

it. T_{seed} is built as a generalization of the objects falling in N_{seed} (function *neighborhoodModel*). When the neighborhood is estimated to be a homogeneous set (function *homogeneity*), cluster C is grown with the structured objects enclosed in N_{seed} which are not yet assigned to any cluster. The cluster C is then iteratively expanded by merging the R-neighborhoods of each node of C (neighborhood expansion) when these neighborhoods result in homogeneous sets with respect to current cluster model T_C (function *expandCluster*, see Algorithm 4.2). T_C is obtained as the set of first-order theories generalizing the neighborhoods merged in C. It is noteworthy that when a new R-neighborhood is built to be merged in C, all the objects which are already classified into a cluster different from C are removed from the neighborhood. When the current cluster cannot be further expanded it is labeled with $CLabel$ and an unclassified seed node for a new cluster is chosen from G until all objects are classified. $CLabel$ is obtained by T_C (function *labelCluster*) to compactly describe C.

This is different from spatial clustering performed by GDBSCAN, although both methods share the neighborhood-based cluster construction. Indeed, GDBSCAN retrieves all objects density-reachable from an arbitrary core object by building successive neighborhoods and checks density within a neighborhood by ignoring the cluster. This yields a density-connected set, where density is efficiently estimated independently from the neighborhoods already merged in forming the current cluster. However, this approach may lead to merge connected neighborhoods sharing some objects but modeling different phenomena. Moreover, GDBSCAN computes density within each neighborhood according to a weighted cardinality function (e.g. aggregation of non spatial values) that assumes single table data representation. CORSO overcomes these limitations by computing density within a neighborhood

in terms of degree of similarity among all relationally structured objects falling in the neighborhood with respect to the model of the entire cluster currently built. In particular, following the suggestion given in [MF03], we evaluate homogeneity within a neighborhood N_{seed} to be added to the cluster C as the average degree of matching between objects of N_{seed} and the cluster model $\{T_C, T_{seed}\}$. Details on cluster model determination, neighborhood homogeneity estimation and cluster labeling are reported below.

4.2.1 Cluster model generation

Let C be the cluster currently built by merging w neighborhood sets $N_1, \ldots, N_w$, we assume that the cluster model T_C is a set of first-order theories $\{T_1, \ldots, T_w\}$ for the concept C where T_i is a model for the neighborhood set N_i. More precisely, T_i is a set of first-order clauses: $T_i : \{cluster(X) = c \leftarrow H_{i1}, \ldots, cluster(X) = c \leftarrow H_{iz}\}$, where each H_{ij} is a conjunctive formula describing a sub-structure shared by one or more objects in N_i and $\forall o_i \in N_i$, $BK \cup T_i \models o_i$. Such model can be learned by resorting to the ILP system ATRE [Mal03] that adopts a separate-and-conquer search strategy to learn a model of structured objects from a set of training examples and eventually counter-examples (see Section 3.4). In this context, ATRE learns a model for each neighborhood set without considering any counter-examples (see Section 3.4.1). The search of a model starts with the most general clause, that is,$cluster(X) = c \leftarrow$, and proceeds top-down by adding selectors (literals) to the body according to some preference criteria (e.g. number of objects covered or number of literals).

Selectors involving both numerical and categorical descriptors are handled in the same way, that is, they have to comply with the property of linkedness and are sorted according to preference criteria. The only difference is that selectors involving numerical descriptors are generalized by computing the closed interval that best covers positive examples and eventually discriminates from counter-examples, while selectors involving categorical descriptors with the same function value are generalized by simply turning all ground arguments into corresponding variables without changing the corresponding function value.

4.2.2 The homogeneity evaluation of sets of objects

In classical clustering methods, objects grouping is performed by exclusively relying on some measures of object similarity. Such measures of similarity are *context free*, in the sense that the similarity between any two objects depends solely on the properties of the two objects involved in the computation and is not influenced by any context. Consequently, methods that use such measures are unable to capture the properties that characterize a cluster as a whole and are not derivable from properties of individual entities.

In order to detect such properties, the system should be equipped with the ability to recognize configurations of objects representing certain global concepts, leading to the basic notion of conceptual clustering where the dominant aim is to find the concepts underlying the objects distribution.

In CORSO we are interested in measuring how much objects arranged into a neighborhood are homogeneous, that is, similar each others. This imposes that similarity evaluation must be performed among all the objects belonging the same group and not calculated between pairs of objects. In other words, we need a unique value of homogeneity associable with a group of objects rather than a number of homogeneity values concerning all the possible pairs of objects. To this aim, our similarity measure is evaluated by generating a model (in term of a logical theory) generalizing all the objects belonging to a given neighborhood and by calculating the average similarity between this model and each object in the set. In this way, when objects are very similar each other, their model will be very close to each of them since it will be described by common features by excluding the different ones and, consequently, the homogeneity value will be high. On the contrary, when objects are dissimilar each other, their model will tend to be quite general since the number of common features will be low, this leading to a low value of homogeneity evaluation.

An example of model generation and homogeneity evaluation performed by CORSO is sketched in Figure 4.2.

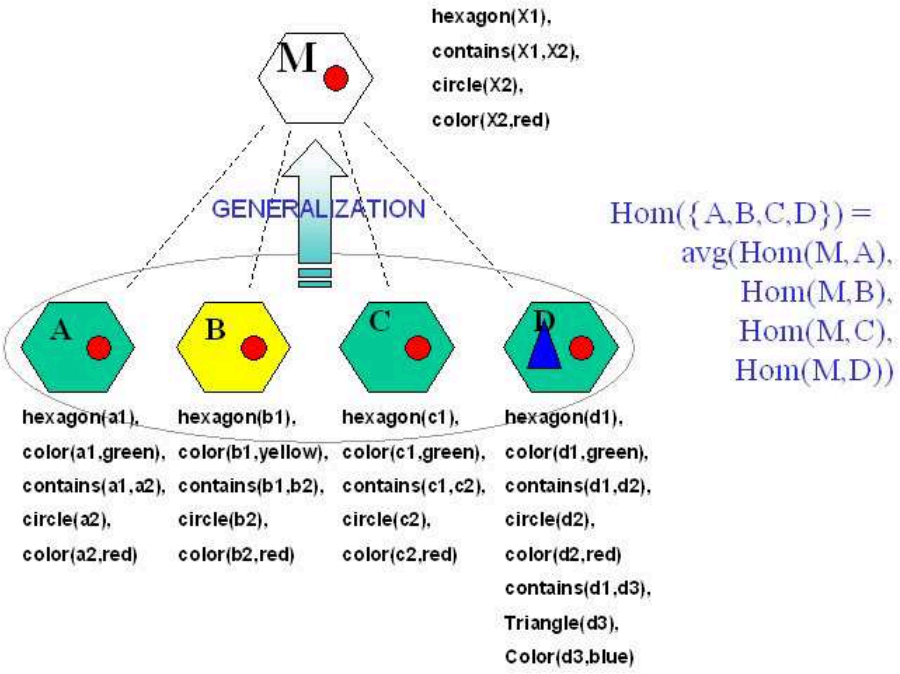


FIGURE 4.2: An example of model generation and homogeneity evaluation performed by CORSO

More formally, the homogeneity of a neighborhood set N_o (with $o \in O$) to be added to the cluster C is computed as follows:

$$h(N_o, T_{C \cup N_o}) = \frac{1}{\#N_o} \sum_{o_i \in N_o} h(o_i, T_{C \cup N_o}) = \frac{1}{\#N_o} \sum_{o_i \in N_o} \frac{1}{w+1} \sum_{T_j \in T_{C \cup N_o}} h(o_i, T_j) \tag{4.1}$$

where $\#N_o$ is the cardinality of the neighborhood set N_o and $T_{C \cup N_o}$ is the cluster model of $C \cup N_o$ formed by both $\{T_1, \ldots, T_w\}$ (i.e., the model of C) and T_{w+1} (i.e., the model of N_o), built as explained in Section 4.2.1. Since $T_j = H_{1j}, \ldots, H_{zj}$ ($z \geq 1$) with each H_{ij} a conjunctive formula in first-order formalism, we assume that:

$$h(o_i, T_j) = \frac{1}{z} \sum_i fm(o_i, H_{ij}) \tag{4.2}$$

where fm is a function returning the degree of matching of an object $o_i \in N_o$ against the conjunctive formula H_{ij}.

In this way, the definition of homogeneity of a neighborhood set $N_o = \{o_1, \ldots, o_n\}$ with respect to some logical theory $T_{C \cup N_o}$ is closely related to the problem of comparing (matching) the conjunctive formula f_i representing an object $o_i \in N_o$[1] with a conjunctive formula H_{ij} forming the model T_j in order to discover likenesses or differences [Pat91]. This is a directional similarity judgment involving a *referent* R, that is the description or prototype of a class (cluster model) and a *subject* S that is the description of an instance of a class (object to be clustered).

In the classical matching paradigm, the matching of S against R corresponds to compare them just for equality. In particular, when both S and R are conjunctive formulas in first-order formalism, matching S against R corresponds to check the existence of a substitution θ for the variables in R such that $S = \theta(R)$. This last condition is generally weakened by requiring that $S \Rightarrow \theta(R)$, where $\Rightarrow$ is the logical implication.

However, the requirement of equality, even in terms of logical implication, is restrictive in presence of noise or variability of the phenomenon described by the referent of matching. This makes necessary to rely on a *flexible* definition of *matching* that aims at comparing two descriptions and identifying their similarities rather than equalities. The result of such a flexible matching is a number in the interval $[0, 1]$ that is the probability of precisely matching S against R, provided that some change described by θ is possibly made in the description R.

The problem of computing *flexible matching* to compare structures is not novel. Esposito et al. [EMS91] have formalized a computation schema for flexible matching on formulas in first-order formalism whose basic components (selectors) are the relational statements, that is, $f_i(t_1, \ldots, t_n) = v$, which are combined by applying different operators such as conjunction ($\wedge$) or disjunction ($\vee$) operator.

In the following, we focus on the computation of flexible matching $fm(S, R)$ when both S and R are described by conjunctive formulas and $fm(S, R)$ looks for the substitution θ returning the best matching of S against R, as:

$$fm(S, R) = \max_{\theta} \prod_{i=1,\ldots,k} fm_\theta(S, r_i). \tag{4.3}$$

where R is assumed to be composed by k selectors $r_1, \ldots, r_k$.

[1] The conjunctive formula f_i is here intended as the description of $o_i \in N_o$ saturated according to the BK.

The optimal θ that maximizes the above conditional probability is here searched by adopting a branch and bound algorithm that expands the least cost partial path by performing quickly on average [EMS91]. According to this formulation, fm_θ denotes the flexible matching with the tie of the substitution fixed by θ computed on each single selector $r_i \equiv f_{r_i}(t_{r_1}, \ldots, t_{r_n}) = v_{r_i}$ of the referent R where f_{r_i} is a function descriptor with either numerical (e.g. area or distance) or categorical (e.g. intersect) range. In the former case the function value v_{r_i} is an interval value ($v_{r_i} \equiv [a, b]$), while in the latter case v_{r_i} is a subset of values ($v_{r_i} \equiv \{v_1, \ldots, v_M\}$) from the range of f_{r_i}. This faces with a referent R that is obtained by generalizing a neighborhood of objects in O. Conversely, for the subject S (that is, the description of a single object $o \in O$), the function value w_{s_j} assigned to each selector $s_j \equiv f_{s_j}(t_{s_1}, \ldots, t_{s_n}) = w_{s_j}$ is an exactly known single value from the range of f_{s_j}.

In this context, the flexible matching $fm_\theta(S, r_i)$ evaluates the degree of similarity $fm(s_j, \theta(r_i))$ between $\theta(r_i)$ and the corresponding selector s_j in the subject S such that both r_i and s_j have the same function descriptor $f_r = f_s$ and for each pair of terms $\langle t_{r_i}, t_{s_i} \rangle$, $\theta(t_{r_i}) = t_{s_i}$. More precisely,

$$fm(s_j, \theta(r_i)) = fm(w_{s_j}, v_{r_i}) = \max_{v \in v_{r_i}} P(equal(w_{s_j}, v)). \tag{4.4}$$

The probability of the event $equal(w_{s_j}, v)$ is then defined as the probability that an observed w_{s_j} is a distortion of v, that is:

$$P(equal(w_{s_j}, v)) = P(\delta(X, v) \geq \delta(w_{s_j}, v)) \tag{4.5}$$

where X is a random variable assuming value in the domain D representing the range of f_r while δ is a distance measure. The computation of $P(equal(w_{s_j}, v))$ clearly depends on the probability density function of X. For categorical descriptors, that is, when D is a discrete set with cardinality $\#D$, it has be proved [EMS91] that:

$$P(equal(w, v)) = \begin{cases} 1 \text{ if } w_{s_j} = v \\ (\#D - 1)/\#D \text{ otherwise} \end{cases} \tag{4.6}$$

when X is assumed to have a uniform probability distribution on D and $\delta(x, y) = 0$ if $x = y$, 1 otherwise. Although similar results have been reported for both linear non numerical and tree-structured domains, no result appears for numerical domains. Therefore, we have extended definitions reported in [EMS91] with the Theorem 4.1 in order to make flexible matching able to deal with numerical descriptors.

Theorem 4.1 *Let S be a subject and R a referent. Let $s_j \equiv f(t_{s_1}, \ldots, t_{s_n}) = c$ be a selector of S and $r_i \equiv f(\theta(t_{r_1}), \ldots, \theta(t_{r_n})) = [a, b]$ be a selector of $\theta(R)$, with θ a substitution for R and f be a numerical descriptor. Then, the flexible matching $fm(s_j, \theta(r_i)) = fm(c, [a, b])$ can be calculated as follows:*

$$fm(c, [a, b]) = \begin{cases} 1 & \text{if } a \leq c \leq b \\ 1 - 2(a - c)/(\beta - \alpha) & \text{if } c < a \wedge 2a - c \leq \beta \\ (c - \alpha)/(\beta - \alpha) & \text{if } c < a \wedge 2a - c > \beta \\ (\beta - c)/(\beta - \alpha) & \text{if } c > b \wedge 2b - c < \alpha \\ 1 - 2(c - b)/(\beta - \alpha) & \text{if } c > b \wedge 2b - c \geq \alpha \end{cases}$$

by assuming that X has uniform distribution on D, α and β are the lower and upper bounds of the interval D respectively, and $\delta(x, y) = |x - y|$.

Proof 4.1 Let us recall definitions (4.4) and (4.5) and apply them to numerical case. We have:

$$fm(c, [a, b]) = \max_{v \in [a,b]} P(equal(c, v)) = \max_{v \in [a,b]} P(\delta(X, v) \geq \delta(c, v))$$

By assuming that X has an uniform distribution on domain $D = [\alpha, \beta]$ with density function $f_D(x) = 1/(\beta - \alpha), \forall x \in D$ and fixing $\delta(x, y) = |x - y|$, $P(\delta(X, v) \geq \delta(c, v))$ can be rewritten as $P(|X - v| \geq |c - v|)$ that is maximized when minimizing $|c - v|$. We can distinguish three cases:

- If $a \leq c \leq b$, then $|c - v|$ is minimized when $v = c$. Hence $\max_{v \in [a,b]} P(|X - v| \geq |c - v|) = P(|X - c| \geq |c - c|) = 1$.

- If $c < a$, then $c < v$ and, hence, $\max_{v \in [a,b]} P(|X - v| \geq |c - v|)$ is written as $\max_{v \in [a,b]} P(|X - v| \geq v - c)$.
  Since the maximum of $P(|X - v| \geq v - c)$ is obtained for $v = a$, we have that $\max_{v \in [a,b]} P(|X - v| \geq v - c) = P(|X - a| \geq a - c) = P(X - a \geq a - c) + P(X - a \leq c - a) = P(X \geq 2a - c) + P(X \leq c)$ where:

  1. $P(X \geq 2a - c) = \int_{\beta}^{2a-c} 1/(\beta - \alpha) dx = \begin{cases} (\beta - 2a + c)/(\beta - \alpha) & \text{if } 2a - c \leq \beta \\ 0 & \text{otherwise} \end{cases}$
  2. $P(X \leq c) = (c - \alpha)/(\beta - \alpha)$.

  Hence, we obtain that:

  $$\max_{v \in [a,b]} P(|X - v| \geq v - c) = \begin{cases} 1 - 2(a - c)/(\beta - \alpha) & \text{if } c < a \wedge 2a - c \leq \beta \\ (c - \alpha)/(\beta - \alpha) & \text{if } c < a \wedge 2a - c > \beta \end{cases}$$

- If $c > b$, then $c > v$ and, hence, $\max_{v \in [a,b]} P(|X - v| \geq |c - v|)$ can be equivalently written as $\max_{v \in [a,b]} P(|X - v| \geq c - v)$. Here, the maximum of $P(|X - v| \geq c - v)$ is obtained for $v = b$. Therefore, $\max_{v \in [a,b]} P(|X - v| \geq c - v) = P(|X - b| \geq c - b) = P(X - b \geq c - b) + P(X - b \leq b - c) = P(X \geq c) + P(X \leq 2b - c)$, where:

  1. $P(X \geq c) = (\beta - c)/(\beta - \alpha)$
  2. $P(X \leq 2b - c) = \int_{2b-c}^{\alpha} 1/(\beta - \alpha) dx = \begin{cases} (2b - c - \alpha)/(\beta - \alpha) & \text{if } 2b - c \geq \alpha \\ 0 & \text{otherwise} \end{cases}$

  In this case, we have that:

  $$\max_{v \in [a,b]} P(|X - v| \geq c - v) = \begin{cases} (\beta - c)/(\beta - \alpha) & \text{if } c > b \wedge 2b - c < \alpha \\ 1 - 2(c - b)/(\beta - \alpha) & \text{if } c > b \wedge 2b - c \geq \alpha \end{cases}$$

□

4.2.3 Cluster labelling

A cluster C can be naturally labelled with T_C that is the set of first-order clauses obtained from the generalization of neighborhoods merged in C. Each first-order clause is in the form $C \leftarrow s_1, \ldots, s_n$, where C represents the cluster label and each s_i denotes a selector in the form $f_i(t_{i_1}, \ldots, t_{i_l}) = v_i$.
In this formalization, two selectors $s_1 : f_1(t_{1_1}, \ldots, t_{1_l}) = v_1$ and $s_2 : f_2(t_{2_1}, \ldots, t_{2_l}) = v_2$ are *comparable* according to some substitution θ when they involve the same descriptor ($f_1 = f_2 = f$) and each pair of terms $\langle t_{1_i}, t_{2_i} \rangle$ is unifiable according to θ, i.e., $t_{1_i}\theta = t_{2_i}\theta = t_i$ $(\forall i = 1 \ldots l)$. In this case, the selector $s : f(t_1, \ldots, t_l) = \{v_1\} \cup \{v_2\}$ is intended as a generalization for both s_1 and s_2.

In particular, the selectors s_1 and s_2 are *equal* when they are comparable and $v_1 = v_2 = v$ such that the generalization of s_1 and s_2 is built as $s : f(t_1, \ldots, t_l) = v$.

Similarly, the selector s_1 (resp., s_2) is *contained* in the selector s_2 (resp., s_1) when they are comparable and $v_1 \subseteq v_2$ (resp., $v_2 \subseteq v_1$), while the generalization s is $f(t_1, \ldots, t_l) = v_2$ (resp., $f(t_1, \ldots, t_l) = v_1$). Note that equality of selectors implies containment, but not vice-versa.

Similarly, the first-order clauses $H_1 : C \leftarrow s_{1_1}, \ldots, s_{1_n}$ and $H_2 : C \leftarrow s_{2_1}, \ldots, s_{2_n}$ are *comparable* according to some substitution θ when each pair of selectors $\langle s_{1_i}, s_{2_i} \rangle$ is comparable according to θ. Hence, H_1 is *equal* (resp., *contained*) to H_2 when s_{1_i} is equal (resp., contained) to s_{2_i} for each $i = 1, \ldots, n$. In both these cases (equality and containment condition), the pair of first-order clauses H_1, H_2 can be replaced without lost of information with the first-order clause H that is the generalization of H_1, H_2 built by substituting each pair of comparable selectors $\langle s_{1_i}, s_{2_i} \rangle \in \langle H_1, H_2 \rangle$ with the generalization obtained as stated above. This suggests the idea of merging a pair of comparable first-order clauses H_1, H_2 in a single clause H whenever the equivalence coverage is preserved, that is:

- for each structured object o with $H_1, H_2, BK \models o$ then $H, BK \models o$ and vice-versa
- for each structured object o with $H_1, H_2, BK \not\models o$ then $H, BK \not\models o$ and vice-versa

where BK is a set of first-order clauses.

The equivalence of coverage between $\{H_1, H_2\}$ and H is obviously guaranteed when H_1 is either equal or contained in H_2 or vice-versa, but this equivalence cannot be guaranteed when H_1 and H_2 are comparable first-order clauses but neither equality condition nor containment condition are satisfied.

Example 1: Let us consider the pair of comparable first-order clauses:

$H_1 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..10], type(X_2) = street$

$H_2 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [3..7], type(X_2) = river$

where neither H_1 is equal to H_2 nor $H_1(H_2)$ is contained in $H_2(H_1)$. The first-order clause obtained by generalizing pairs of comparable selectors in both H_1 and H_2, is $H : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [3..10], type(X_2) = \{street, river\}$, where $H \models o$ with $o : distance(X_1, X_2) = 3 \wedge type(X_2) = street$, but neither $H_1 \models o$ nor $H_2 \models o$.

The requirement of equality between H_1 and H_2 can be relaxed while preserving equivalence of coverage with respect to the generalization H. Indeed, when

$H_1 : C \leftarrow s_1(\_) = v_1, \ldots s_k(\_) = v_k, \ldots, s_n(\_) = v_n$

$H_2 : C \leftarrow s_1(\_) = v_1, \ldots s_k(\_) = w_k, \ldots, s_n(\_) = v_n$

are comparable first-order clauses differing only in the function value of a single selector (i.e. s_k), the first-order clause:

$H : C \leftarrow s_1(\_) = v_1, \ldots s_k(\_) = \{v_k\} \cup \{w_k\}, \ldots, s_n(\_) = v_n$

continues to preserve the equivalence of coverage with $\{H_1, H_2\}$.

Example 2: Let us consider the pair of comparable first-order clauses:

$H_1 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [3..7], type(X_2) = street,$
$length(X_2) = [3, 5]$

$H_2 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [3..7], type(X_2) = street,$
$length(X_2) = [7, 10]$

which differ only in the value of a single selector (length), the first-order clause obtained by generalizing the pairs of comparable selectors in both H_1 and H_2 is:

$H : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [3..7], type(X_2) = street,$
$length(X_2) = [3, 5] \cup [7, 10]$

that is equivalent in coverage to the pair $\{H_1, H_2\}$.

Following this idea, it is possible to compactly describe the cluster theory T_C finally associated to a cluster C by iteratively replacing pairs of comparable first-order clauses H_1, H_2 with the generalization H, when H results equivalent in coverage to $\{H_1, H_2\}$ (see Algorithm 4.3).

Example 3: Let us consider T_C that is the set of first-order clauses including:

$H_1 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..10], color(X_2) = red$

$H_2 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..6], color(X_2) = blue$

$H_3 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..10], color(X_2) = blue$

$H_4 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [6..10], area(X_2) in [30..40]$

T_C can be transformed in the set of first-order clauses:

$H'_1 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..10], color(X_2) = \{red, blue\}$

$H'_2 : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [6..10], area(X_2) in [30..40]$

where H'_1 results by firstly merging H_1 and H_3, which are comparable and differ only in the function value of a selector ($color(X_2) = red$ vs $color(X_2) = blue$), and obtaining $H_{13} : cluster(X_1) = c \leftarrow distance(X_1, X_2) = [5..10], color(X_2) = \{red, blue\}$ and then merging H_{13} and H_2 since H_2 is contained in H_{13}.

4.3 The seed selection problem

The cluster shapes depend on the object that CORSO chooses at each step as seed of the neighborhood to be considered. As basic approach, CORSO adopts a sequential strategy. In the case a new cluster has to be discovered, the seed is the first object accessed in O not yet assigned to any cluster. In the case an existing cluster has to be expanded, the seed is the object stored in the first position of a list collecting the cluster objects not yet considered for the expansion step.

Algorithm 4.3 Build a compact theory to describe a cluster C.

```
1: function clusterLabel(T_C) → T'_C;
2: T'_C ← ∅
3: merge ← false;
4: while T_C is not empty do
5:     H is a first-order clause in T_C;
6:     T_C = T_C/H;
7:     for each H' ∈ T_C do
8:         if H and H' are generalizable without lost of information then
9:             H = generalize(H,H'); T_C = T_C/H'; merge = true;
10:        end if
11:    end for
12:    T'_C = T'_C ∪ H;
13: end while
14: if merge is true then
15:    T'_C ←clusterLabel(T'_C);
16: end if
17: return T'_C;
```

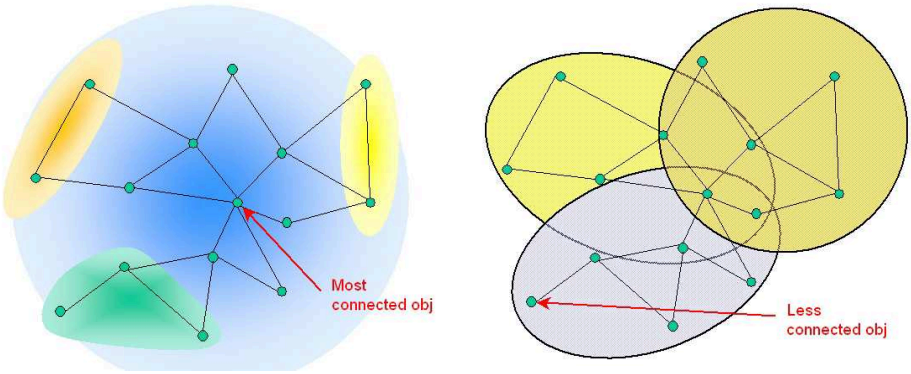
The sequential seed selection (SEQ) is efficient, but quality of clustering clearly depends on the "order" of storing and accessing objects, which is not necessarily the best one. For this reason, two alternative strategies have been empirically investigated and implemented in CORSO that take advantage from the graph structure in seed selection step. In particular, we base the choice of the candidate seed on the concept of density (cardinality) of a neighborhood.

In each seed selection, the "best" candidate seed is the object whose neighborhood in the graph has the highest density (or the first one according to the descending order of connectivity, *DESC*) or the lowest density (or the first one according to the ascending order of connectivity, *ASC*).

In the first case (DESC), we follow the intuition coming from the density-based framework where dense areas are labelled as clusters. Our suggestion is to estimate density in terms of the number of connections in the graph from the candidate seed to its neighboring objects. According to this strategy, the objects to prefer in the seed selection are the most connected ones because in general social and environmental phenomena affecting a spatial region (such as unemployment rates in population or pollution distribution over territory) are more prominent in the centre of the area and tend to soften in peripheric zones (see Figure 4.3.a).

In the second case (ASC), the basic assumption is that a dense area in the graph may correspond to an area covered by contiguous different clusters, hence, discovery should preferentially start from peripheral objects. In this strategy the less connected objects are preferred in the seed selection since the underlying idea is that objects with a high number of connections are also affected by a higher number of phenomena characterizing surrounding areas (see Figure 4.3.b).

Currently, CORSO implements all the three strategies presented above for the



(a) Spatial arrangement of phenomena (b) Phenomena overlapping in central areas

Figure 4.3: Motivations of descending (a) and ascending (b) order of connectivity in the seed selection

seed selection: sequential selection (SEQ), ascending order of connectivity (ASC) and descending order of connectivity (DESC). However, as future work, the method could be extended by a further strategy that puts together all the above mentioned strategies. In particular, the idea is to calculate the clustering results coming from the three approaches and select that one that is best scored according to some heuristic that evaluates the clusters quality.

4.4 Improving CORSO

As stated earlier in this chapter, CORSO builds clusters by merging partially overlapping neighborhoods, which are *homogeneous* according to the homogeneity function h. However, the neighborhood-based evaluation as defined in the previous section suffers from several problems that have been noticed in the application of the system and overcome as follows.

Firstly, the homogeneity is estimated at the *neighborhood level*. This means that a cluster as well as its description is "incrementally" built without guaranteeing that the entire cluster results to be homogeneous with respect to its final description. To avoid this incoherence, we have modified CORSO in order to build clusters by merging contiguous neighborhoods, but evaluating homogeneity at *cluster level*. The neighborhood N_o is merged to the cluster C if and only if the homogeneity of the candidate cluster $C_i = C \cup N_o$ is greater than a user-defined threshold. This, in practice, changes the test performed by the system when deciding if to expand a cluster with a neighborhood or not:

$$\begin{aligned} \text{old evaluation test: } C_i = C \cup N \quad & iff \quad h(N, T_{C_i}) > threshold \\ \text{new evaluation test: } C_i = C \cup N \quad & iff \quad h(C_i, T_{C_i}) > threshold \end{aligned} \tag{4.7}$$

Homogeneity of C_i is evaluated as:

$$h(C_i, T_{C_i}) = \frac{1}{\#C_i} \sum_{o_i \in C_i} fm(o_i, T_{C_i}), \tag{4.8}$$

where $T_{C_i} = \{T_C, T_{N_o}\}$. Since homogeneity evaluation at cluster level is more complex than evaluation at neighborhood level, some caching techniques are applied to improve scalability of the algorithm, resulting in the following equation:

$$h(C_i, T_{C_i}) = \frac{\frac{\#C}{\#T_{C_i}} \cdot (\#T_C \cdot h(C, T_C) + \#T_{N_o} \cdot h(C, T_{N_o})) + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i} \tag{4.9}$$

where $C_i - C$ represents the set of objects obtained by removing objects in C from C_i (set subtraction).

The equation 4.9 has been calculated as follows:

$$h(C_i, T_{C_i}) = \frac{1}{\#C_i} \sum_{o_j \in C_i} h(o_j, T_{C_i}) = \frac{1}{\#C_i} \sum_{o_j \in (C + (C_i - C))} h(o_j, T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} h(o_j, T_{C_i}) + \frac{1}{\#C_i} \sum_{o_j \in (C_i - C)} h(o_j, T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} h(o_j, T_C \cup T_{N_o}) + \frac{\#(C_i - C)}{\#C_i} \cdot \frac{1}{\#(C_i - C)} \sum_{o_j \in (C_i - C)} h(o_j, T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} (\frac{1}{\#T_{C_i}} \cdot \sum_{H_k \in (T_C \cup T_{N_o})} fm(o_j, H_k)) + \frac{\#(C_i - C)}{\#C_i} \cdot h((C_i - C), T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} (\frac{1}{\#T_{C_i}} \cdot (\sum_{H_k \in T_C} fm(o_j, H_k) + \sum_{H_k \in T_{N_o}} fm(o_j, H_k))) +$$

$$+ \frac{\#(C_i - C)}{\#C_i} \cdot h((C_i - C), T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} (\frac{1}{\#T_{C_i}} \cdot (\frac{\#T_C}{\#T_C} \cdot \sum_{H_k \in T_C} fm(o_j, H_k) + \frac{\#T_{N_o}}{\#T_{N_o}} \cdot \sum_{H_k \in T_{N_o}} fm(o_j, H_k))) +$$

$$+ \frac{\#(C_i - C)}{\#C_i} \cdot h((C_i - C), T_{C_i}) =$$

$$= \frac{1}{\#C_i} \sum_{o_j \in C} (\frac{1}{\#T_{C_i}} \cdot (\#T_C \cdot h(o_j, T_C) + \#T_{N_o} \cdot h(o_j, T_{N_o}))) + \frac{\#(C_i - C)}{\#C_i} \cdot h((C_i - C), T_{C_i}) =$$

$$= \frac{\sum_{o_j \in C} (\frac{1}{\#T_{C_i}} \cdot (\#T_C \cdot h(o_j, T_C) + \#T_{N_o} \cdot h(o_j, T_{N_o}))) + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i} =$$

$$= \frac{\frac{\sum_{o_j \in C} (\#T_C \cdot h(o_j, T_C) + \#T_{N_o} \cdot h(o_j, T_{N_o}))}{\#T_{C_i}} + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i} =$$

$$= \frac{\frac{\#T_C \cdot \sum_{o_j \in C} h(o_j, T_C) + \#T_{N_o} \cdot \sum_{o_j \in C} h(o_j, T_{N_o})}{\#T_{C_i}} + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i} =$$

$$= \frac{\frac{\#T_C \cdot \frac{\#C}{\#C} \cdot \sum_{o_j \in C} h(o_j, T_C) + \#T_{N_o} \cdot \frac{\#C}{\#C} \cdot \sum_{o_j \in C} h(o_j, T_{N_o})}{\#T_{C_i}} + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i} =$$

$$= \frac{\frac{\#C}{\#T_{C_i}} \cdot (\#T_C \cdot h(C, T_C) + \#T_{N_o} \cdot h(C, T_{N_o})) + \#(C_i - C) \cdot h((C_i - C), T_{C_i})}{\#C_i}$$

Consequently, $h(C, T_C)$ values can be stored as basis for future evaluations.

Secondly, the homogeneity is estimated with respect to a cluster description (T_{C_i}) that is a set of first-order theories, one for each neighborhood merged to form the candidate cluster C_i. Originally, the theory for each neighborhood was learned independently from the theory currently associated with the cluster to be expanded. Hence, the same theory can be learned for separate neighborhoods by introducing duplicates in the evaluation of homogeneity. This can be avoided by changing the cluster model induction in order to take into account the current cluster description T_C when generalizing objects of a potential neighborhood. Practically, learning is performed by only considering the neighbors of N_{o_i} not covered by T_C itself. In this way, efficiency of the cluster evaluation is improved since CORSO avoids to recompute the homogeneity of an object with respect to the same theory (labeling several neighborhoods) many times. In addition, clusters are naturally labeled with more compact descriptions which are simpler to be interpreted.

Thirdly, the clustering expansion operates at neighborhood level. This prevents the expansion of a cluster by adding only a "portion" of a neighborhood so forcing the shape of discovered clusters. To avoid this problem, the system is improved by performing a single-neighbor based expansion when failing the neighborhood-based one. Whenever a neighborhood N is estimated to be not homogeneous with respect to the cluster in expansion C (because its homogeneity value is less than the considered threshold), then each single object o_N belonging to N is tried to be added to C and is definitively added to it if $C \cup \{o_N\}$ is still homogeneous.

The usefulness of this additional test is highlighted in the example depicted in Figure 4.4.

Here, the cluster C cannot be expanded with the neighborhood N because the set candidate to become a cluster, $C \cup N$, is not homogeneous. Such heterogeneity is due to the presence of grapevines in N. Nevertheless, N also contain a red apple that is perfectly compliant with the model of the cluster C. The only way to add this object in C is to perform a single-neighbor based check as described above that, consequently, adds more accuracy in the shape of detected clusters.

In addition to the above mentioned improvements, another desirable ability of a clustering method could be the discovery of (partially) overlapped clusters other than the detection of partitions of the set of objects O. However, discovering overlapping clustering or, in other words, associating each spatial object to one or

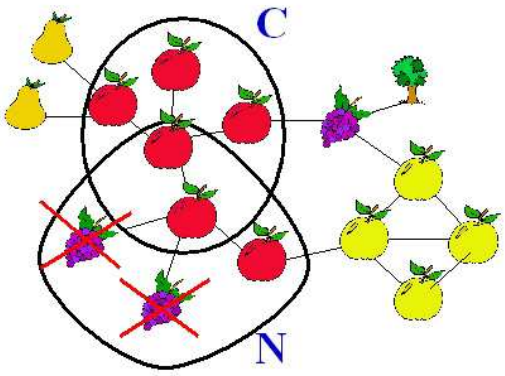


FIGURE 4.4: Single-neighbor based expansion of clusters

more different clusters, is a more computationally expensive task than discovering partitions of data because each object needs to be examined as many times as the number of detected clusters.

In order to be able to discover partially overlapping clusters without having a substantial downfall in its efficiency, CORSO implements an additional optional step at the end of clustering detection that is devoted to detect possible areas where clusters overlap. During this step, each cluster is examined to find all the boundary objects (a boundary object is characterized to be also linked with objects not belonging to the same cluster). Each external object (either a noise object or an object belonging to a neighboring cluster) linked with a boundary object is tried to be added to the current cluster. If it results to be homogeneous to the current cluster, then it is added to the cluster and the test is propagated to its neighbors; otherwise, no action is undertaken and another boundary object is chosen.

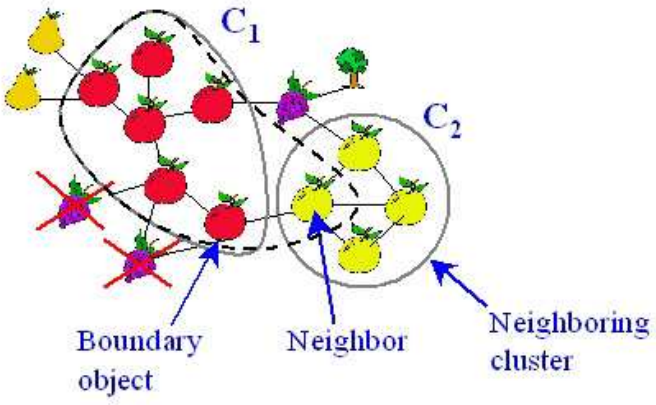


FIGURE 4.5: Overlapping cluster detection

In the example depicted in Figure 4.5, the lowest red apple of cluster C_1 is a boundary object for C_1 since it is linked with a yellow apple, say it o, belonging to

the cluster C_2. In this case, the homogeneity value of $C_1 \cup \{o\}$ is estimated and, if the homogeneity test successes, o is added to C_1 (as a consequence, o belongs to both clusters C_1 and C_2) and the test is performed on the new neighbors.

4.5 Customizing the algorithm: parameters settings

Sometimes in literature it is stated that a desirable property of clustering algorithms is the presence of few or even no parameters which values need to be specified by the user [Kol01, Ber02]. The motivation is that the less are the required parameters, the less are the additional information that user must provide to the system to calculate clusters. Since user generally does not know in advance such information (or does not have a clear idea about them) and since this information is used by the system in the search of the solution, input parameters are considered to be a quite strong limitation for clustering methods because a wrong setting of such parameters heavily affects the clustering results and even can prevent the system to find a solution.

However, clustering methods without input parameters guiding the search in the hypothesis space result to be quite rigid in the sense that they could not be profitably suited to the needs of both the working domain and the learning task to be solved. As a simile, clustering methods can be compared to radios: a radio with only a volume control and a tuning knob is certainly easy to use even for non-expert users, but a radio that also has an equalizer can be used both by a beginner user with default settings and by an expert user that can use the equalizer to adapt the radio behaviour to his/her own needs.

In this perspective, the optimal solution is to follow the general suggestion of reducing as much as possible the mandatory parameters to be specified by the user, but leaving room for customization that is useful in specific domains or where domain knowledge can be useful to guide the learner. As a consequence, input parameters with default values can represent the right trade-off between simplicity of use and method flexibility.

Following this idea, CORSO makes use of only one mandatory parameter (specifying the file containing the input data to be clustered) and nine optional parameters, as described below:

- *input*: the name of the input file (in XML format) containing data to be clustered; this parameter is mandatory
- *output*: the name of the output file (in XML format) resulting from a clustering task; if not specified, the name of the output file is composed as "<inputxmlfile>_output.xml", where "<inputxmlfile>" is the name of the input file
- *logFile*: the name of a valid text file where the log information are saved; if not specified, the log file is not written
- *threshold*: the threshold value to be used in the evaluation of neighborhoods homogeneity; the default value for this parameter is 0.90

- *server*: the (logical or IP) address of the remote machine where ATRE Server is running; the default value is "localhost"
- *seedselection*: the strategy adopted in the seed selection; valid values are "seq" (objects are read sequentially), "aoc" (ascending order of connectivity) and "doc" (descending order of connectivity); the default value is "seq"
- *punctualAdding*: enables (if set to "true") or disable (if set to "false") the single-neighbor cluster expansion whenever a neighborhood results to be heterogeneous with respect to a given threshold; the default value is "true"
- *obtainOverlap*: enables (if set to "true") or disable (if set to "false") the detection of (partially) overlapping clusters; the default value is "false".

Other than the previous mentioned parameters, two further parameters can be set to decide the granularity to which the neighborhoods must be calculated.

The neighborhood-based exploration of available data requires a high number of steps to discover clusters when phenomena to be modelled affect large portions of space. In order to detect such large clusters in a lower number of steps and, hence, to improve the efficiency of the method, CORSO allows user to change the granularity in the neighborhood detection. Usually, the neighborhood of a given object o is represented by the set of objects directly linked with o according to the considered relation. We can widen the set of considered objects by reiterating n times the detection of neighborhoods. In this way we can observe the presence of different levels of granularity for the neighborhoods, where the term "*granularity*" indicates the extent of calculated neighborhoods: fixing a seed object, a neighborhood of a given granularity is completely enclosed in a neighborhood of a greater granularity. In general, a neighborhood at level n (with $n \geq 1$) is composed by the objects belonging to the paths of length n in the graph G (see Section 4.2) starting from the considered seed object o. For instance, for $n = 2$, the neighborhood at level 2 $N_2(o)$ is composed by the set of objects directly linked to o (neighborhood at level 1, $N_1(o)$) and the set of objects linked with the objects in $N_1(o)$, while for $n = 3$ the neighborhood at level 3 $N_3(o)$ is composed by objects belonging to $N_2(o)$ plus the objects directly linked to these ones (see Figure 4.6).
In general, for $k > 1$:

$$N_k(o) = N_{k-1}(o) \cup \{o_j \in O | \text{ there is an object } o_i \in N_{k-1}(o) \text{ and an arc linking } o_i \text{ to } o_j \text{ in } G\}$$

These neighborhoods can be examined to search for a homogeneous set of objects according two directions: from the innermost towards the outermost or vice-versa. This behaviour can be defined by means of the following parameters:

- *maxrange*: the maximum neighborhood level to be considered in the clustering task; the default value is 1, that means no reiterations are allowed for the neighborhoods detection
- *growing*: considers the ascending order (if set to "true"), that is from 1 to *maxrange*, or the descending order (if set to "false"), that is from *maxrange* to 1 in the calculation of reiterated neighborhoods; the default value is "true".

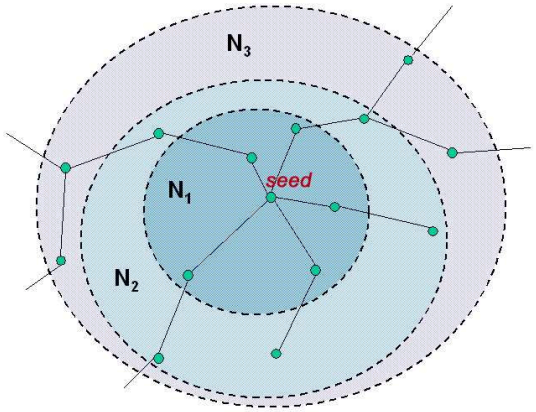


Figure 4.6: The granularity levels in the neighborhoods detection

4.6 Complexity analysis

This section is devoted to calculate the computational complexity of performing a clustering task with CORSO.

Computational complexity is aimed to calculate what is the computational effort or resources (either in space or in time) needed to perform a clustering task. To make this calculation easier to perform, only the dominant operations (considering the complexity and the quantity) performed during each task execution are taken into account. For instance, the computational complexity of a program calculating the factorial function is determined by only considering the product operations performed during the task.

As concerns CORSO, complexity is calculated by considering the cost of either accessing an object from DB or performing a learning task for model induction, since these two operations are considered the more expensive from the computational point of view by a long way.

For sake of simplicity, we assume that the considered neighborhoods, both in creation and in expansion phase, are composed by all the neighbors, without leaving out the objects already assigned to a different cluster.

Let $|O| = n$ be the cardinality of the set of spatial objects to be clustered and l the average number of links for each node (this meaning that each spatial object is related with l different objects according to the considered relation R).
Let us distinguish three cases:

BEST CASE: all the considered spatial objects are homogeneous, that is, CORSO groups all the objects in a cluster ($|C| = 1$ and $C = O$).
The number of steps performed in the clustering task is $n - l$, because at $(n - l)$-th step all the objects are assigned to the cluster. Moreover, at each clustering step $l+1$ objects are examined, namely the seed object and its l neighbors. Consequently, by considering the DB accesses, the number of accesses to the DB is $(n - l)(l + 1)$, therefore, taking into account that the dominant variable (that assumes the highest

values) is n, the computational complexity is $O(n)$.
By considering the induction of neighborhood models, the number of the learning tasks performed is 1, since the system detects only one cluster.

WORST CASE: all the considered spatial objects are heterogeneous, that is, CORSO labels all the input objects as noise ($|C| = 0$).
The number of steps performed in the clustering task is n, because no seed is found by the system having a homogeneous neighborhood and, hence, giving rise to a cluster. Furthermore, at each clustering step $l + 1$ objects are examined, namely the seed object and its l neighbors. As a consequence, the number of accesses to the DB is $n(l+1)$, so the computational complexity by considering the DB accesses is $O(n)$.
By considering the number of induction tasks performed to detect the neighborhood models, the complexity is estimated to be n, since CORSO checks the homogeneity of n neighborhoods in the search for cluster.

AVERAGE CASE.
Let us assume that the number of clusters to be detected is m. Consequently, the average number of objects per cluster is n/m.
For each cluster, other than the access to the initial seed, $(n/m) - 1$ expansion attempts are performed since an expansion attempt is performed for each object in the neighborhood except for the seed. In addition, for each expansion attempt $l + 1$ objects are examined (the seed object and its neighbors). Since the number of clusters is assumed to be m, the number of accesses to the DB is equal to $m + m((n/m) - 1)(l + 1) = nl + n - ml$. For this reason, the computational complexity by taking into account the DB accesses is again $O(n)$.
Concerning the number of learning tasks performed to generate the neighborhood models, by assuming that half of the expansion attempts lead to have neighborhoods where at least one object exists that is not covered by the cluster model (case in which a learning task is required to try to expand the cluster model), the complexity is equal to $m+m((n/m)-1)(1/2) = m+(n/2)-(m/2) = (2m-m+n)/2 = (m+n)/2$.

4.7 Representing the clustering results

Generally, in data mining most of attention is concentrated on the aspects concerning the extraction of novel and useful knowledge from raw data, stored in databases or in other storage formats and architectures. However, the huge effort employed by the scientist in the definition of more and more complex and effective methods can be unfruitful if few attention is paid in the representation of results. The quality of results obtained by applying a clustering method can be appreciated and, hence, really used only if their representation allows the user to highlight, as easily as possible, the essence of mined patterns.

In the context of spatial clustering problem solved by CORSO, resulting clusters should be presented in a way that highlights both the spatial arrangement of data

and their structural resemblances. In addition, it is also important to let the user to navigate through clustered data in order to focus on some particular zones of interest, or to have an overview of data as a whole, or to move from an area to another. To this aim, CORSO relies on three different forms of clusters representation:

- textual representation
- graph-based representation
- map-based representation

The *textual representation* is the most immediate and simple of those available. It does not provide any information about the spatial arrangement of data or the relations existing among them, but shows the detected clusters (and the objects belonging to each of them) into a tree-based structure. In this way users can easily browse the tree structure to explore the list of clusters. For each cluster, this view makes available the intensional description (that is, the cluster model), the extensional description (that is, the list of objects belonging to it) other than some simple statistical information like the number of objects and the rate of data assigned to the cluster (see Figure 4.7).

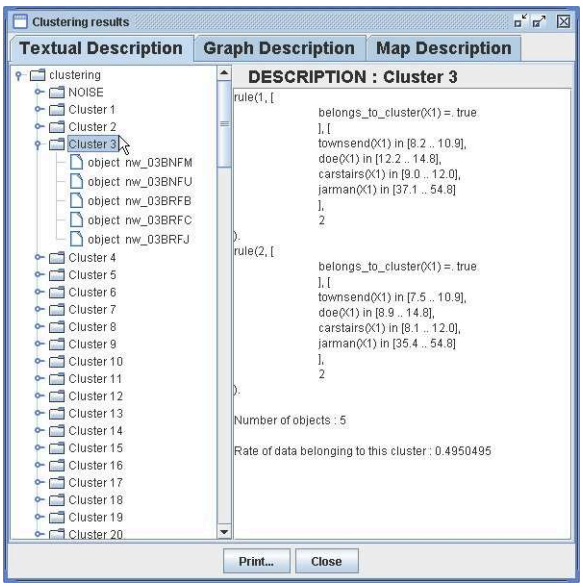


Figure 4.7: The textual representation of CORSO results

The *graph-based representation* allows user to see and navigate the graph resulting from the discrete spatial structure according to which data are arranged into space. This view displays the membership of data to detected clusters and the relations holding among them, but cannot show the geometrical information of data such as shapes and territory coverage.

Membership of nodes to clusters is intuitively represented by the color of each represented node: a different color is associated to each different cluster and this color association is used to paint the nodes on the screen. The cluster colors are

automatically calculated by the system but user can easily customize the cluster-color associations by simply clicking on the button labelled with the name of the cluster and selecting a different color. In order to ensure a different color for each cluster even when the number of clusters is huge, the RGB components of each color are calculated as follows:

- RED component (**R**): $(|255 + ((-1)^i) \cdot 23 \cdot i|) mod (255)$
- GREEN component (**G**): $(|255 + ((-1)^i) \cdot 13 \cdot i|) mod (255)$
- BLUE component (**B**): $(|255 + ((-1)^i) \cdot 53 \cdot i|) mod (255)$

where i is the cluster position in the list of the clusters (for instance, $i = 0$ is the first detected cluster, $i = 1$ is the second one, and so on). It is easy to see how the three RBG components of each color are created according to the pattern

$$(|255 + ((-1)^i) \cdot p \cdot i|) mod (255)$$

with p be a prime. Here, the goal is to obtain different colors for each iteration, and this can be obtained by calculating RBG components as distant as possible. To this aim, for each iteration, the RGB components are calculated by alternating the sign of the value $(p \cdot i)$ added to 255. Primes (p) are used to avoid to obtain the same value for two different iterations i_1 and i_2, while distant primes are used in each RGB component formula in order to use different steps for each color component, thus avoiding to obtain gray colors. The obtained numbers are reported in the range [0..255], that is, the range of valid values. In addition, when geometry information are available for the current dataset, this representation paints each node with a different radius according to the size of the corresponding spatial object.

Other than the visual representation of the graph of data clustered by CORSO, when user selects an object this view also displays the object identifier and the identifier and the model of the cluster assigned to it (see Figure 4.8).

The last and the more powerful view available in CORSO system is the map-based representation. This view highlights the geometrical information of clustered data showing, other than the cluster membership of objects by means of the color assignment (like in the graph-based representation), the shapes of the objects in the dataset and, hence, producing a rendering of the map of the territory where clustering has been applied (see Figure 4.9).

Differently from the graph-based representation discussed above, that is centred on spatial relations linking objects, this form of representation hides the relations among the spatial objects and focusses the attention on how the detected clusters make a partition of the territory. This results in a data perspective that could be very useful in decision making frameworks.

In conclusion, since the geometrical information represented in the last two representations are not present in the XML file resulting from the clustering task, a further XML file containing geometry information is necessary to enable both the graph-based and the map-based representation. This XML file contains the list of objects in the dataset describing, for each of them, the coordinates of the centroid, the size and the shape described in a formalism similar to that one adopted by the SDO_GEOMETRY object type of Spatial Oracle 10g [Ora05].

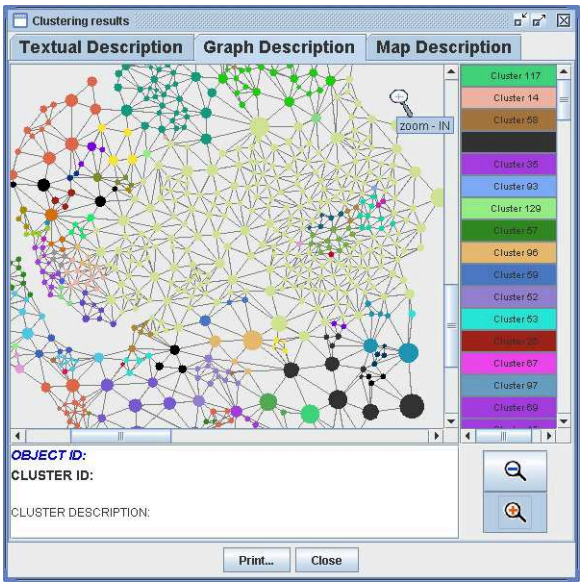


Figure 4.8: The graph-based representation of CORSO results

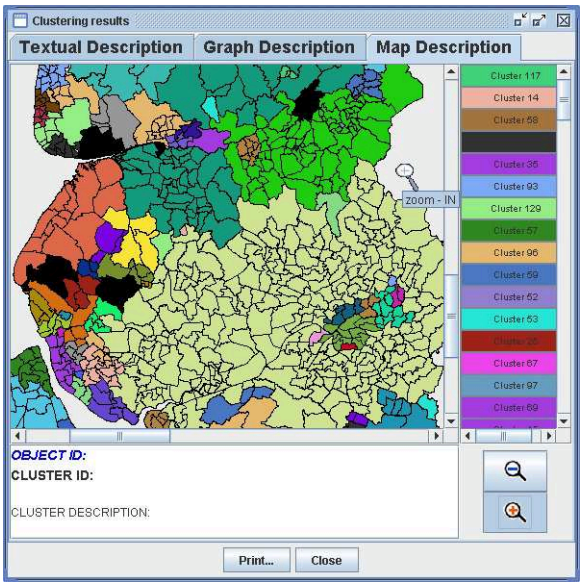


Figure 4.9: The map-based representation of CORSO results

4.8 Conclusions

In this chapter we have presented CORSO, a spatial clustering method that overcomes limitations of both spatial clustering and multi-relational approach. We have shown how CORSO detects clusters of objects by taking into account the spatial correlation of data and their structural resemblances. Spatial correlation of data is represented by links among nodes corresponding to spatial objects in a graph-like structure where data are arranged, while structural resemblance is detected by relying on a homogeneity evaluation function expressing how much objects in a neighborhood are similar each other. CORSO builds clusters by putting together objects that are both linked and similar each other.

Complexity study reported in Section 4.6 shows that CORSO takes a time linearly dependent with the number of objects to be clustered.

In addition, issues concerning the seed selection are discussed and some solution have been proposed and implemented in the system. Finally, some improvement of the method have been presented such as the ability of detecting (partially) overlapping clusters and representation techniques adopted by CORSO are reported at the end of the chapter.

Chapter 5

Applications

Spatial clustering can be profitably applied in a large number of human fields such as social science, remote sensing, bioinformatics, etc. However, most of applications concern geographical contexts, where the geographical entities (cities, for instance) and their relations (such as the presence of a state road connecting cities) can be naturally mapped with nodes and edges of a discrete spatial structure.

In this chapter, we present some applications of CORSO method, some of them working on artificial data and some others on real datasets.

The first experiment, presented in Section 5.1, works on artificial data and is aimed at discovering groups of similar computers connected each others in a LAN. The dataset has been deliberately built to have irregularly-shaped clusters and nested clusters, that is, situations where a cluster is completely contained into another one. The goal of the experiment is to examine the behaviour (in terms of obtained results) of the method over such a particular configuration of data. Moreover, CORSO is compared with other similar clustering methods in order to highlight resemblances and differences in the corresponding results.

Section 5.2 deals with the application of spatial clustering to a novel field that is gathering increasing interest of scientist in recent years: Social Network Analysis. In particular, the system is applied to a network of people working in the same company in order to discover groups of similar persons that should work together. Even if this experiment can be applied to real data, the dataset dealt here is artificial for two main reasons: (i) this experiment is a preliminary study for the application of CORSO to such domain and (ii) it is necessary to preserve the privacy of people involved in the study so, to avoid privacy infringement, individuals and data are unreal.

The third section (Section 5.3) presents the application of our method in a territory characterization problem. The experiment concerns an area located in Apulia region and aims at applying the clustering method presented in this dissertation to partition the territory into homogeneous areas. In order to evaluate if and how system performance is affected by the dimension of data to be clustered, different runs have been performed of the system on this dataset by varying the number of objects to be clustered and the number of features describing each object.

Section 5.4 describes the application of CORSO to real data concerning the North-West England territory. Since each county in the dataset is described by both geomorphological features and social indicators, the expected results are supposed to relate social factors to geographical characterization of areas.

Finally, application of CORSO to air pollution data in Apulia region is discussed in Section 5.5. Air pollution is an increasing problem that is gathering more and more interest by public authority. Data have been gathered by means of fixed or mobile air pollution testing stations belonging to three different networks. However, such air pollution testing stations are irregularly scattered over the territory and this represent a heavy limitation to a good partition of the region on the basis of pollutants.

5.1 Experiments on artificial data: the LAN dataset

In this experiment the goal is to apply clustering to artificial data describing computers arranged into a Local Area Network (LAN). The arrangement of personal computers composing the dataset is described in Figure 5.1. The task is to find groups of similar computers connected each others.

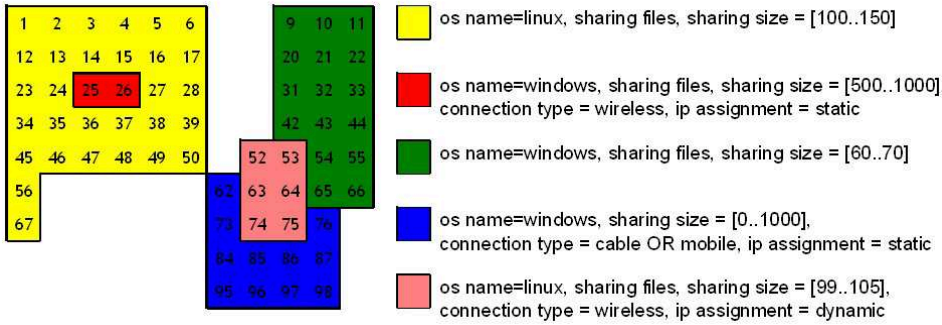


Figure 5.1: The LAN dataset

Each structured object is identified by a natural number and graphically represented in the figure with a square. It corresponds to a PC described in terms of operative system used, shared option, shared space size, type of net connection and type of IP assignment, while the discrete spatial structure is defined by the adjacency relation. PC are arranged into five different clusters. Each cluster is associated with a description and it is identified by a different color. For instance, the yellow cluster describes the set of contiguous computers having Linux as o.s. and sharing an amount of HD space ranging from 100 MB to 150 MB. The arrangement of data has been artificially generated by including irregularly shaped (green and blue) and/or concentric (red and yellow) clusters.

CORSO has been applied to this dataset by using different parameters configurations (see Section 4.5 for details on CORSO customization) in order to investigate and discuss the parameters setting that gives the best results. In addition, both

the initial and the improved version of CORSO have been applied with the aim of comparing the results obtained with the neighborhood-based and the cluster-based homogeneity evaluation function adopted in clusters detection.

Results obtained with the initial version of the system (neighborhood-based evaluation of cluster homogeneity) are reported in Figure 5.2.

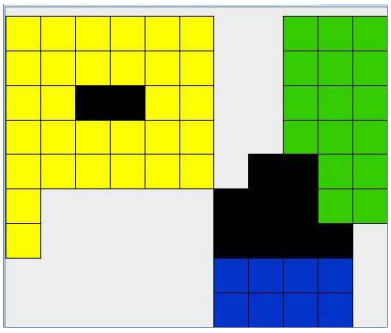


FIGURE 5.2: Clusters obtained by performing a neighborhood-based homogeneity evaluation (threshold = 0.95)

The above figure shows that the system performs moderately well on this dataset: two out of five clusters (the yellow and green clusters) are perfectly detected discovering their precise shapes, while the blue cluster is detected with an almost exact shape. Only the red and pink clusters of Figure 5.1 have not been discovered by CORSO. This happens because the objects 25 and 26 are surrounded by elements dissimilar with respect to them (belonging to the yellow clusters), so the homogeneity value of these neighborhoods are less than the specified threshold. The same motivation holds for the objects of the pink cluster, which neighborhoods are not homogeneous because contain objects belonging to the green or blue clusters. In all, the system labels as noise 11 out of 65 objects (about 17% of data).

Looking at the cluster models, the system has been able to discover the features characterizing the clusters, even if a number of repetitions occurs in the mined models. For instance, the model associated to the yellow cluster is composed by the following clauses:

$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [100..150]$ *(8 occurrences)*
$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [110..150]$ *(3 occurrences)*
$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [100..125]$ *(2 occurrences)*
$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [110..145]$
$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [100..135]$
$belongsToCluster(X1) = true \leftarrow osName(X1) = linux, sharingFiles(X1) = true, sharedSize(X1) \in [100..140]$

Such a redudancy in models is due to neighborhood descriptions in cluster expansion phases that are performed without taking into account the description of the cluster in expansion, as described in the discussion of Section 4.4.

By applying the version of CORSO adopting a cluster-based homogeneity evaluation of neighborhoods, the quality of results clearly improve. CORSO results obtained with different parameter configurations are shown in Figure 5.3 (homogeneity threshold is set to 0.95). In this study, we have investigated the cluster-based homogeneity function in combination with the sequential (SEQ), ascending (ASC) and descending (DESC) order of connectivity for the seed selection as well as the cluster expansion performed at single-neighbor level (SNE).

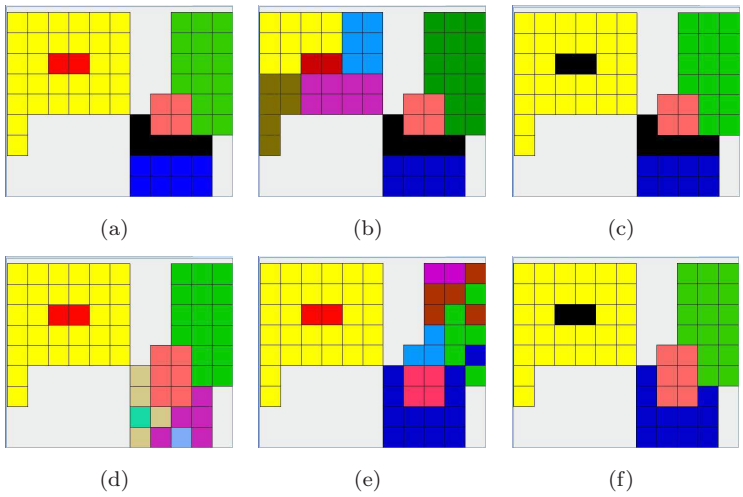


Figure 5.3: Clusters obtained with the cluster-based homogeneity evaluation in different configurations: sequential (a), ascending (b) and descending (c) order for seed selection. In the second row, results obtained with sequential (d), ascending (e) and descending (f) order for seed selection and single-neighbor check in cluster expansion are shown.

Results confirm that, except for the ASC configuration, the quality of clusters is improved with respect to the first version of the system since the shape of clusters better fits the original one reported in Figure 5.1. At the same time, the number of objects labeled as noise (black objects) decreases from 11 in the initial version of CORSO (neighborhood-based evaluation function and sequential seed selection) to 5 in SEQ and ASC configurations, 7 in DESC configuration when resorting to the cluster-based homogeneity evaluation. The ASC configuration detects a higher number of clusters as an evidence that preferring less connected objects in the seed selection generally leads to more fragmented results. This is a consequence of the fact that peripheral objects are generally affected by neighboring clusters smoothing the discontinuity of some phenomenon (first Law of Geography, [Tob79]). In particular, the original yellow cluster shown in Figure 5.1 is fragmented into four separate clusters (see Figure 5.3.b), but a deeper analysis reveals that the descriptions associated with these clusters are quite similar:

yellow cluster:
$\ldots \leftarrow osName(X) = linux, sharingFiles(X) = true, sharedSize(X) \in [100..150]$
brown cluster:
$\ldots \leftarrow osName(X) = linux, sharingFiles(X) = true, sharedSize(X) \in [100..135]$
fuchsia cluster:
$\ldots \leftarrow osName(X) = linux, sharingFiles(X) = true, sharedSize(X) \in [100..150]$
azure cluster:
$\ldots \leftarrow osName(X) = linux, sharingFiles(X) = true, sharedSize(X) \in [100..125]$.

The clusters descriptions discovered with cluster-based homogeneity are simpler than those discovered with the neighborhood-based homogeneity evaluation. For example, the description of the yellow cluster in Figure 5.2 is composed by 16 similar (or at worst identical) clauses, while the same description includes only one clause in the SEQ configuration (Figure 5.3.a). This depends on the fact that the cluster-based homogeneity evaluation re-uses the current cluster description when building the generalization of a candidate neighborhood thus avoiding to introduce redundant clauses in the final cluster description.

Finally, results reported in Figures 5.3.d,e,f confirm that the cluster expansion performed at single-neighbor level allows to discover irregularly shaped clusters. In fact, CORSO with SEQ seed selection is able to perfectly discover the pink cluster even if some errors are performed in detecting the blue one (see Figure 5.3.d). Still, the worst results are obtained with the ASC seed selection, although no object is labeled as noise. On the contrary, the DESC seed selection produces the best result, that is, it perfectly detects the actual clusters except for objects 25 and 26 that are labelled as noise during the expansion of contiguous clusters.

Additionally, CORSO has also been applied to this dataset by enabling the discovery of multi-cluster objects (see Sections 4.4 and 4.5 for further details). Figure 5.4 depicts the results of this run, where the threshold value has been fixed to 0.95 and the *Descending Order of Connectivity* has been used as seed selection criterion.

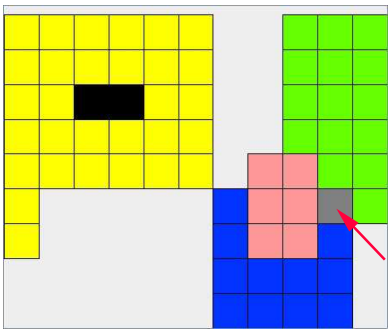


FIGURE 5.4: Clusters obtained by performing a neighborhood-based homogeneity evaluation (threshold = 0.95; seed selection criterion = doc)

Figure 5.4 shows that, in comparison with the results reported in Figure 5.3.f, the discovery of multi-cluster objects has led to detect an overlapping area (composed

by the gray object and highlighted by the red arrow) between the green and blue clusters. Indeed, the gray object represents a pc with the following features:

os_name = windows
connection_type = mobile
sharing_files = true
sharing_size = 70
ip_assignment = static

that really belongs to both the green cluster (representing computers with Windows O.S. and that share files with an amount of shared size between 60 and 70) and blue cluster (representing computers with Windows O.S., having a static IP assignment type and that share files with an amount of shared size between 0 and 1000).

Other than CORSO, a number of clustering tasks have been performed on this dataset by using different methods well-known in literature and enclosed in Weka system, version 3.5.0 [WF00, Wek05, KF05], with the aim of comparing results obtained with CORSO and clusters calculated with the considered methods. Weka is a comprehensive tool that encompasses different Data Mining algorithms (classification, clustering or association rules discovering) and supports user in every step of knowledge flow, including visualization of results. In this experiment, we have performed runs on the K-Means and Expectation Maximization (partitioning methods described in Section 2.2.1), DBScan (density-based method described in Section 2.2.3) and Cobweb (conceptual clustering method described in Section 3.5) algorithms as the major representatives of spatial and conceptual clustering.

Most of the considered Weka algorithms are spatial and, hence, they group objects by only relying on distance measures (Euclidean distance, in general) defined on numeric attributes. Hence, we need to assign spatial coordinates to arrange objects in the space since we cannot rely on the relation among them and the corresponding discrete spatial structure.

In addition, each of considered Weka clustering algorithms needs to be run with specific parameter settings. K-Means has been executed by setting the number of clusters to be discovered (N) equal to 5. This parameter has been chosen by counting the number of natural clusters composing the dataset (but in real cases this number is seldom known in advance). Expectation Maximization has been run by specifying the maximum number of iterations equal to 100, the minimum allowable standard deviation equal to 10^{-6} and leaving unspecified the number of clusters. Concerning the density-based clustering algorithms, we have considered DBScan instead of its generalization GDBScan since generalizations introduced by the latter one does not effectively aid in a best formalization of the relations among spatial data because of its inability to deal with categorical data. DBScan requires to specify two input parameters: ϵ (the radius of the area bounding the neighborhood of an object) and $MinPts$ (the minimum number of neighbors of a core object). In order to choose the best configuration of such parameters, different runs have been performed by setting $MinPts$ equal to 3, 4 and 6, and ϵ equal to 0.9 and 1.03. The best results have been obtained for $\epsilon = 1.03$ and $MinPts = 6$, where the clusters are most similar to the expected ones. Finally, Cobweb has been executed

by specifying two input parameters: *acuity* (the minimum standard deviation for numeric attributes) and $cutoff$ (the category utility threshold by which to prune nodes). In our experiment, the chosen values for these parameters are $acuity = 0.70$ and $cutoff = 0.25$.

However, even when introducing spatial coordinates for considered objects and specifying additional input parameters to guide the clusters search, both spatial and conceptual algorithms perform bad and they result to be not able to mine the real shape or description of clusters, as depicted in figure 5.5.

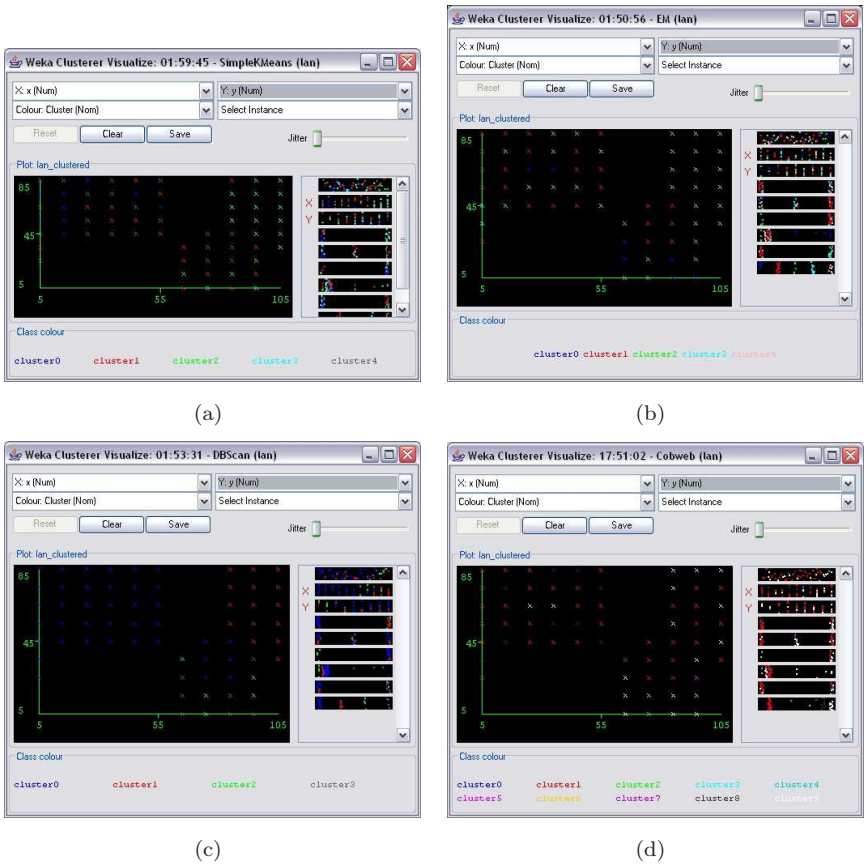


FIGURE 5.5: Results of applications of different clustering methods to the LAN dataset. The figures depicts clusters calculated by K-Means (a), Expectation Maximization (b), DBScan (c) and Cobweb (d) methods.

This happens because all the considered algorithms only focus on either spatial arrangement of data or the concept behind the distribution of data attributes. This enforces the appreciation of CORSO since, differently from the existing algorithms, considers at the same time both the spatial and conceptual aspects of data, being in this way able to extract the real essence of clusters. This capacity is fundamental in cases like this one, where the (structural) resemblance of data is important but needs to be supported by the spatial distribution of data.

5.2 Application of CORSO to Social Network Analysis

Social network analysis (SNA) is the mapping and measuring of relationships and flows between people, groups, organizations, animals, computers or other information/knowledge processing entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. SNA, thanks to its graph-based representation of data, provides both a visual and a mathematical analysis of human relationships.

Understanding networks and their participants implies to evaluate the location of actors in the network. Measuring the network location is finding the *centrality* of a node. These measures give us insight into the various roles and groupings in a network, revealing where are the connectors, mavens, leaders, bridges, recluses, where are the clusters and who is in them, who is in the core of the network, and who is on the periphery.

Usually, SNA has been applied in the context of sociological researches and business management. In companies, coordination and work increasingly occur through informal networks of relationships rather than through channels tightly prescribed by formal reporting structures or detailed work processes. These networks often promote organizational flexibility, innovation and efficiency as well as quality of products or services by virtue of effectively pooling unique expertise. However, even if people rely very heavily on their network of relationships to find information and solve problems, informal networks often compete with and are fragmented by some aspects of organizations as formal structure, work processes, geographic dispersion, human resource practices, leadership style and culture [CBP02]. In addition, informal relationships are often invisible or at least only partially understood by managers, that are responsible of taking decisions. Application of SNA to business management analysis makes it possible to discover and analyse such informal networks.

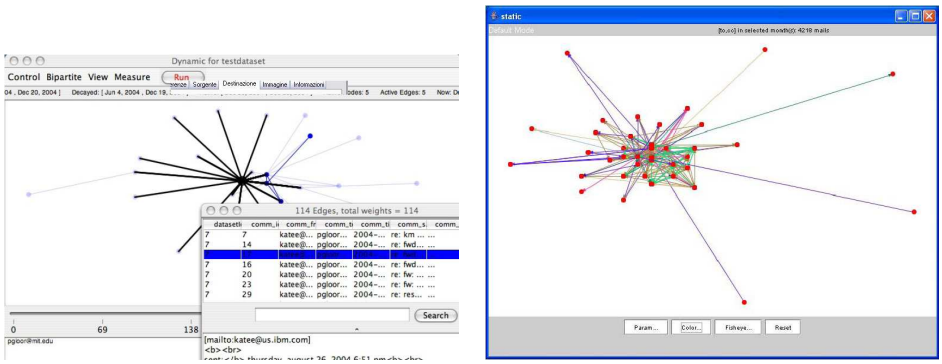
As an alternative scenario, the tragic events of September 11, 2001 have given in recent years particular rise to the application of SNA to uncloak terrorist networks [Kre01]. Here, the goal is to improve national security and fight terrorism by analysing large amounts of data gathered by confidential or public documents in order to discover regularities and rules identifying potential groups of terrorists.

Different tools have been developed by scientific communities or software developers that support officers and personnel managers in SNA studies, such as iQuest [Jai06] or Condor/TeCFlow [GZ05].

IQuest takes as input Outlook or Eudora mailboxes, web mailing lists, on-line forums, web links and flat files. It parses those documents, stores them in a MySQL database and visualizes and analyses this data in manifold interactive and visual ways in order to make easier for the analyst to discover important patterns over data (Figure 5.6.a).

Similarly, TeCFlow takes as input e-mail archives and automatically generates static and dynamic visualizations of the calculated communication networks. The

exchanges of e-mails between actors are considered as an approximation of social ties. Data are represented as a graph: a communication initiated by actor A to actor B is represented as a direct edge from A to B. The more interactions between actors A and B occur, the closer the two representing vertices are placed (Figure 5.6.b).



(a) (b)

FIGURE 5.6: Screenshots of two SNA tools: iQuest (a) and TeCFlow (b).

However, available SNA tools only represent the informal network status according to different perspectives facilitating user in grasping the network structure and related patterns but do not apply DM techniques for the automatic discovery of such patterns. Consequently, CORSO can be profitably applied in SNA domain to search for patterns that could suggest important information about the behaviour of a company and the hidden structures according to it works.

This experiment aims at applying the CORSO spatial clustering technique in a SNA framework in order to evaluate the ability of the system of extracting useful and hidden patterns over data that could helpfully support officers in taking decisions.

The goal is to automatically group employees of a firm according to both their backgrounds and skills and the most frequent contacts between them. This could improve the organization and the productivity of a firm because the system could suggest useful arrangement of employee desks in order to minimize the time waste in internal communications.

As explained above, CORSO could be integrated as an additional feature in the Condor/TeCFlow by using the graph (representing people and communications between them) produced by this system to suggest the groups of employees that should be placed close each others. Obviously, the integration is not trivial and requires a special version of CORSO that should be equipped with a preprocessing module for the generation of input data gathering information from both the graph provided by Condor/TecFlow and from DB (for instance, describing the skills and information concerning employees).

The dataset, described by means of a graph representation, has been artificially built and is composed by 30 employees working in a hypothetical firm. Each em-

ployee works on at least one out of three projects carried out by the firm (*agrifood*, *e-tourism*, *e-banking*) and is represented in the graph by a node. The complete list of people considered in the dataset and their roles are reported in Table 5.1.

Role	**Employee Name**
project-leader	betty
	laura
	lynn
analyst	albert
	criss
	julia
	kevin
	pamela
designer	burt
	max
	michelle
	nicole
	paul
	tom
	vincent

Role	**Employee Name**
developer	alex
	anthony
	ann
	cindy
	daniel
	francis
	george
	john
	kate
	kim
	mark
	mary
	peter
	robert
	steve

Table 5.1: SNA dataset: employees and roles

Concerning the nodes structure, the literals composing the description of each employee are the following:

- *doing_activity* (e.g., doing_activity(kate, analysis)=true)
- *used_tool* (e.g., used_tool(bob,eclipse)=true, used_tool(bob,jbuilder)=true, used_tool(kate,rational_rose)=true)
- *involved_project* (e.g., involved_project(jack,e_tourism)=true)
- *performing_skill* (e.g., performing_skill(kate,case_uses_design)=true, performing_skill (bob,java_programming)=true)
- *need* (e.g., need(jack)=printer, need(kate)=wireless)
- *age* (e.g., age(bob)=28, age(kate)=24)
- *project_leader* (e.g., project_leader(kate)=tom)
- *department* (e.g., department(kate)=R&D)
- *past_experience* (e.g., past_experience(kate,java_programming)=true)
- *type_of_activity* (e.g., type_of_activity(analysis)=theoretical, type_of_activity(developing)=practical, type_of_activity(design)=mixed)
- *before_of* (e.g., before_of(analysis,design)=true)

- *task_in_activity* (e.g., task_in_activity(case_use_design,analysis)=true)
- *tool_for_task* (e.g., tool_for_task(eclipse,java_programming)=true)

Furthermore, each employee can be frequently in contact with his/her colleagues by different means (e-mail, phone, face to face, chat, etc.). In the graph, an edge connects two nodes if the corresponding employees are frequently in contact. For instance, the example depicted in Figure 5.7 describes an object ($p1$) representing an employee (*betty*) that is regularly in contact with employees $p4$ and $p5$, that works on the e-tourism project, that is coordinator and leader of the project at the same time, that currently is responsible for the scheduling activities, being 43 years old, working in the innovation lab and having past experience in class-diagrams writing.

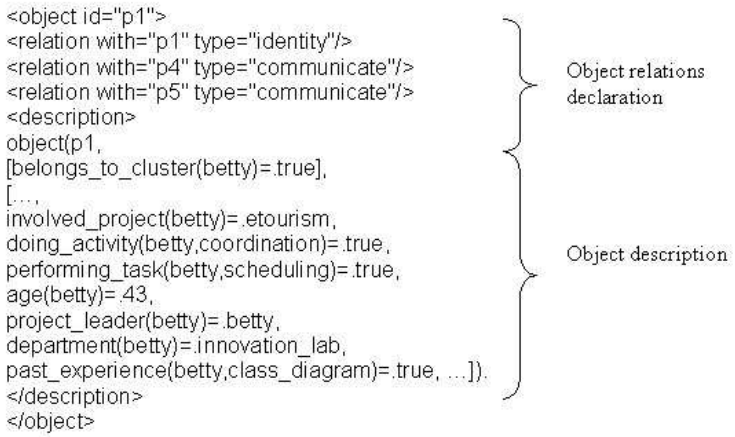


FIGURE 5.7: An example of object description in SNA domain

The graphical representation of the entire dataset, enclosing employees and relations among them, is reported in Figure 5.8.

Three CORSO runs have been performed in this experiment, each of them adopting a different strategy in the seed selection for the clusters detection but a common setting for the other parameters:

- $threshold = 0.99$
- $punctualAdding = true$
- $obtainOverlap = false$
- $maxrange = 1$
- $growing = true$

In the following, the results corresponding to the three applications of CORSO are depicted in Figures 5.9, 5.10 and 5.11. In these figures, each detected cluster has been represented with a different color while gray objects are labelled as *noise* (not assigned to any cluster).

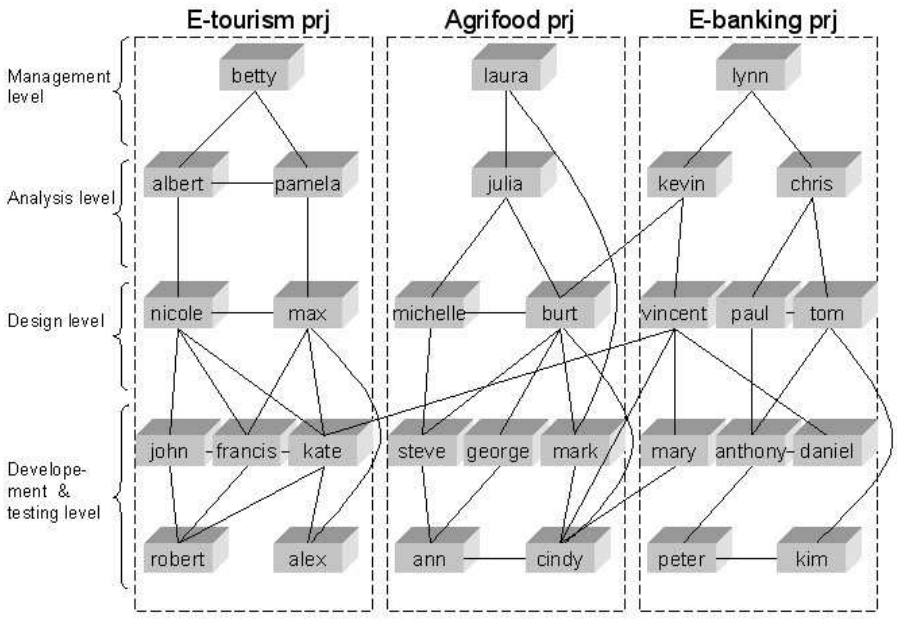


Figure 5.8: SNA dataset: graphical representation of data

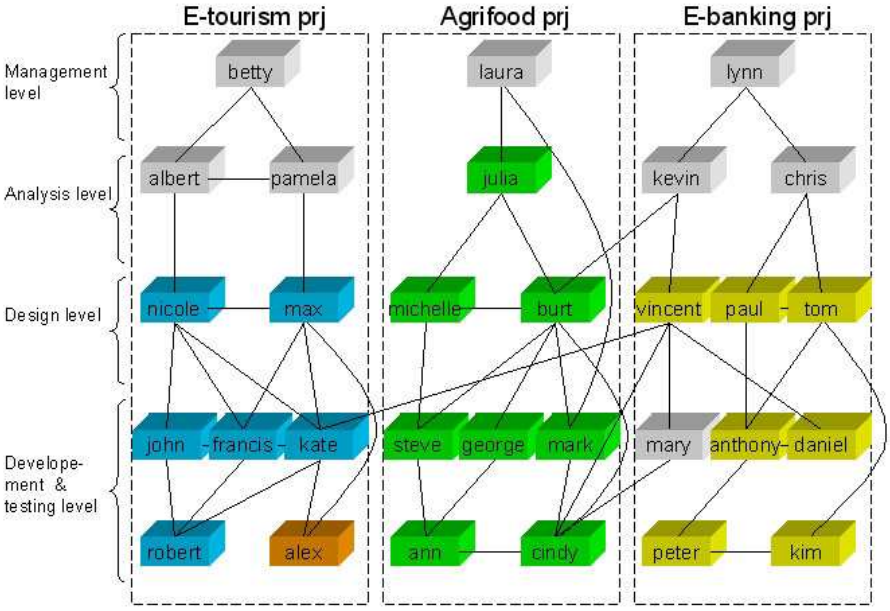


Figure 5.9: Application of CORSO to SNA dataset with a sequential selection of seed objects

As can be easily observed in Figure 5.9, the sequential selection of seed objects leads to label as noise the top levels of the organization chart (management and analysis levels of the three projects). This happens because the learning system deems heterogeneous the activities performed by the management layer and the analysis one. In fact, the system selects as seed the project managers, that are almost exclusively in touch with the analysts of their projects. By looking at the descriptions of project managers and their analysts (for instance, *betty*, *albert* and *pamela* for the e-tourism project), it can be observed that they differs in the type of activity (*managerial* versus *theoretical*) other than the tasks performed (*supervision* and *organize_meetings* performed by managers versus *case_use_design*, *requirement_analysis* and *e_r_analysis* performed by analysts).

Four clusters have been detected with the sequential strategy in the seed selection, each of them representing a group of homogeneous workers. For instance, the blue cluster is described by the following clause:

$belongs\_to\_cluster(X1) = true \leftarrow involved\_project(X1) = etourism,$
$doing\_activity(X1, X2) = true, type\_of\_activity(X2) = practical,$
$activity(X2) = develop, performing\_task(X1, X3) = true,$
$task(X3) = php\_programming$

This means that it is composed by people involved in the e-tourism project, performing a practical activity (software development, in particular) and coding in PHP.

The complete descriptions of the remaining three discovered clusters are reported in the following:

green cluster:

$belongs\_to\_cluster(X1) = true \leftarrow involved\_project(X1) = agrifood,$
$doing\_activity(X1, X2) = true, type\_of\_activity(X2) = practical,$
$activity(X2) = .develop$

yellow cluster:

$belongs\_to\_cluster(X1) = true \leftarrow involved\_project(X1) = ebanking,$
$doing\_activity(X1, X2) = true, type\_of\_activity(X2) = practical,$
$activity(X2) = develop, before\_of(X3, X2) = true,$
$project\_leader(X1) = lynn, department(X1) = production\_lab$

orange cluster:

$belongs\_to\_cluster(X1) = true \leftarrow involved\_project(X1) = etourism,$
$doing\_activity(X1, X2) = true, type\_of\_activity(X2) = practical,$
$activity(X2) = develop$

Notice that the orange cluster is composed by only one objects, since when that object is selected as seed, its neighborhood is empty because all surrounding objects have already been labelled.

By adopting the ascending/descending order of connectivity in the seed selection, CORSO application to this dataset results in the clusters depicted in Figures 5.10

and 5.11.

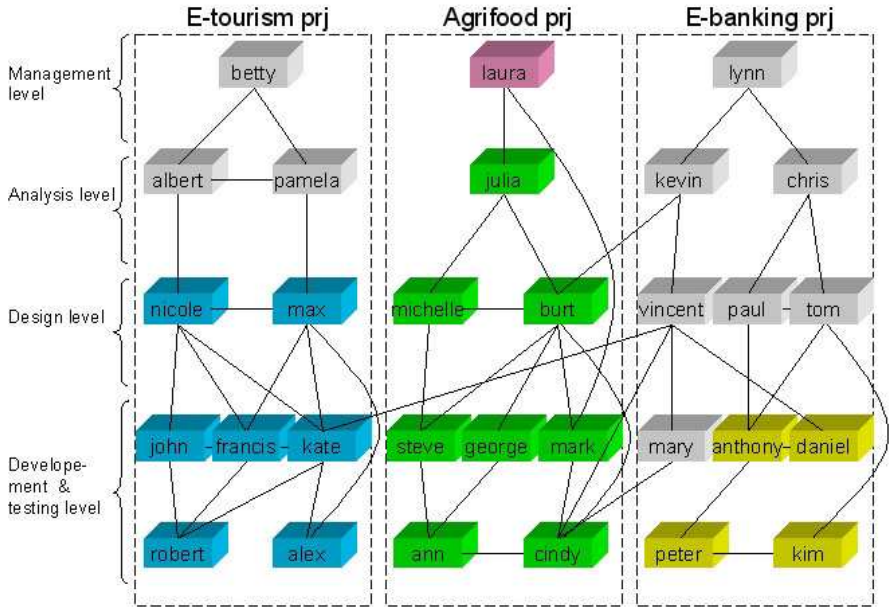


Figure 5.10: Application of CORSO to SNA dataset adopting the ascending order of seed selection

It is straightforward to see how the results obtained by changing the seed selection criterion are quite similar each others. Slight differences can be observed between different runs due to different objects considered as starting points for the cluster detection. In particular, the highest number of objects labelled as noise is obtained with the adoption of the ascending order of connectivity in the seed selection. This confirms the intuition that less connected objects, being peripheral objects in clusters, are less representative of modelled phenomenon and are more heavily affected by surrounding phenomena leading to a more fragmented partition of dataset or to a higher number of noise objects due to the heterogeneity of their neighborhoods. The lowest number of objects labelled as noise is obtained with the adoption of descending order of connectivity in the seed selection, indicating that generally the best criterion to adopt is those that selects the more connected object since it is more likely to be the most representative of the cluster to detect.

5.3 Clustering agricultural zones in the land of Canosa di Puglia

This section is devoted to describe a real-world application of clustering with the aim of partition the geographical territory corresponding to an area in the Apulia region (Italy) according to some distinguishing geomorphological features.

In this case, each structured object is represented by a square region of the map, that is treated as a grid of cells of the same size and is described in terms of geometrical and topological features concerning geographical objects in the territory

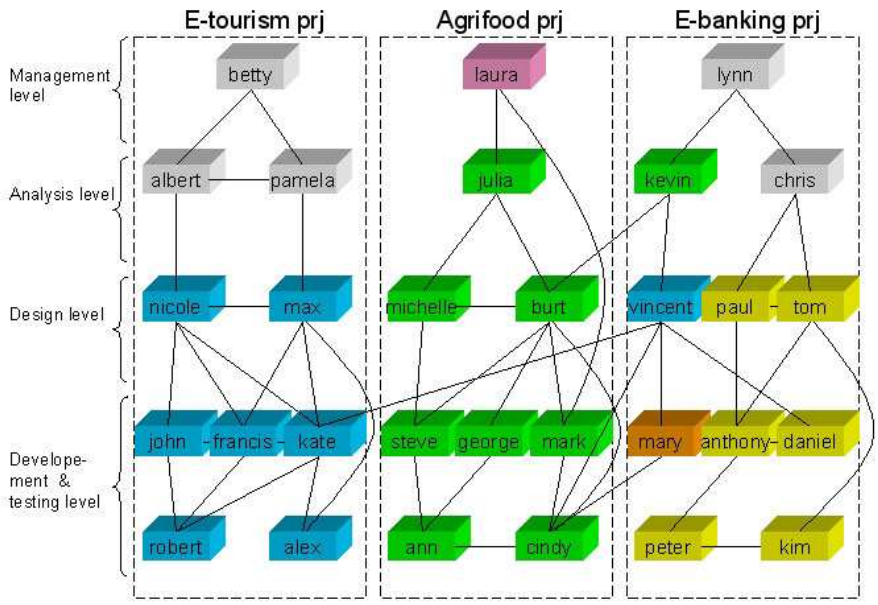


FIGURE 5.11: Application of CORSO to SNA dataset adopting the descending order of seed selection

such as roads, cultivations, fonts, rural areas, etc. The discrete data structure is defined by adjacency relations among cells, so only objects corresponding to cells sharing a side are considered to be related each others.

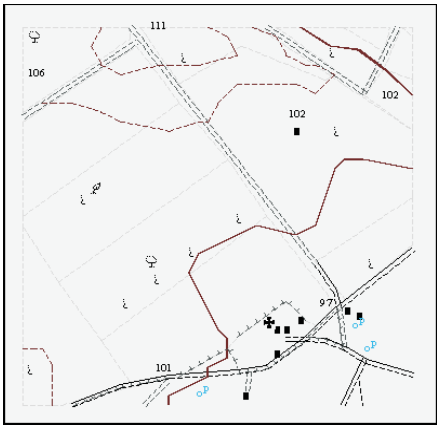
For each cell, geographical data is represented as humans perceive it in reality, that is, geometric (or physical) representation and thematic (or logical) representation. Geometric representation describes the geographical objects by means of the most appropriate physical entity (point, line or region), while thematic representation expresses the semantics of geographical objects (e.g., hydrography, vegetation, transportation network and so on), independently of their physical representation. Spatial clustering aims at identifying a mosaic of homogeneous clusters (areas) including adjacent regions in the map such that geographical data inside each cluster properly models the spatial continuity of some morphological environment within the cluster, while separate clusters model spatial variation over the entire space taken into account.

The territory considered in this application covers 45 km^2 from the zone of Canosa in Apulia. The examined area is segmented into 45 square areal units of 1 Km^2 each (territory partition into cells is depicted in Figure 5.12). Thus, the problem of recognizing spatial continuity of some morphological elements in the map is reformulated as the problem of grouping adjacent cells resulting in a morphologically homogeneous area, that is, a problem of clustering spatial objects according to the discrete spatial structure imposed by the relation of "adjacency" among cells. Since several geographical objects, eventually belonging to different layers (e.g., almond trees, olive trees, fonts, streets, etc.) are collected within each cell, we apply algorithms derived from geometrical and topological reasoning [MEL+03] to ob-

1	2	3	4	5
11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45
51	52	53	54	55
61	62	63	64	65
71	72	73	74	75
81	82	83	84	85

Figure 5.12: The basic partition of Canosa territory into cells

tain cell descriptions in first-order formalism (see Figure 5.13). For this task, we consider descriptions including spatial descriptors encompassing geometrical properties (*area*, *extension*) and topological relations (*regionToRegion*, *lineToLine*, *pointToRegion*) as well as non spatial descriptors ($typeOf$, $subtypeOf$).



contain(c, f2) = true, …,
contain(f, f70) = true,
type_of(c) = cell, …,
type_of(f4) = vegetation,…,
subtype_of(f2) = grapewine,…,
subtype_of(f7) = cart_track_road,…,
part_of(f4, x4),
part_of(f7, x5), part_of(f7_x6),…,
extension(x7) = 111.018,…,
extension(x33) = 1104.74,
line_to_line(x7, x68) = almost_parallel, …,
point_to_region(x4, x21) = inside,
point_to_region(x4, x18) = outside,…,
line_to_region(x8, x27) = adjacent, …

Figure 5.13: First-order description of a cell extracted from topographic chart of Apulia.

As first step of experimentation, CORSO has been applied to the dataset by using the following execution parameters:

- $threshold = 0.99$
- $seedselection = seq$
- $punctualAdding = true$
- $obtainOverlap = false$

This means that in this experiment objects are considered to be homogeneous only when the flexible matching value (fm) between them and their model is equal or greater than 0.99 (that is, by using an almost exact matching). In addition, a

sequential strategy is used in the seed selection (objects are chosen as seeds in the same order they are read) and the single-neighbor cluster expansion is performed when a cluster cannot be further expanded with the classical neighborhood cluster expansion. The capability of discovering (partially) overlapping clusters has been disabled since the goal is to find a partition of the considered space.

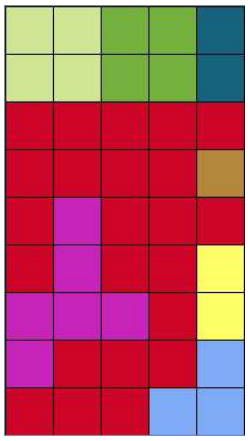


FIGURE 5.14: Resulting clusters for the Ofanto dataset

Figure 5.14 depicts the clusters resulting from the application of CORSO to the Ofanto dataset by using the parameter configuration described above. The eight clusters detected by CORSO are described by the following logical models:

pale-green cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [1..1],$$
$$contains\_font(X1, X2) = true, grapevines\_in\_cell(X1) \in [10..19].$$

green cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [1..4],$$
$$olives\_in\_cell(X1) \in [1..4], grapevines\_in\_cell(X1) \in [9..27],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1102],$$
$$street\_to\_parcel(X2, X3) = adjacent.$$

blue cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [1..2],$$
$$olives\_in\_cell(X1) \in [1..3], grapevines\_in\_cell(X1) \in [6..7],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [22..1044].$$

red cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$$
$$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$$

brown cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [24..24],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [25..862],$$
$$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [1025..66181].$$

fuchsia cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow contains\_street(X1, X2) = true,$$
$$extension\_street(X2) \in [149..1153], street\_to\_parcel(X2, X3) = adjacent,$$
$$area\_growing(X3) \in [2400..132137].$$

yellow cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [3..6],$$
$$olives\_in\_cell(X1) \in [3..4], grapevines\_in\_cell(X1) \in [18..30],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [54..1230].$$

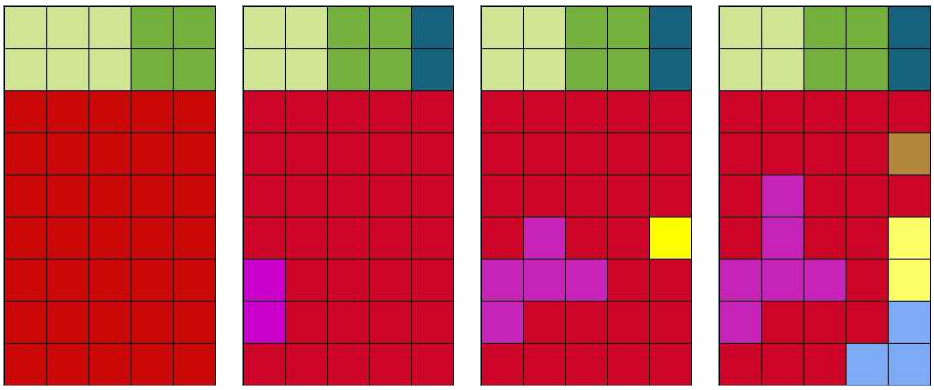
azure cluster:

$$belongs\_to\_cluster(X1) = true \leftarrow contains\_olives(X1, X2) = true,$$
$$olives\_in\_cell(X1) \in [1..6], grapevines\_in\_cell(X1) \in [18..26].$$

Each cluster model clearly describes the main features characterizing the corresponding area encompassed in the cluster boundary, while separate clusters describe quite different environments. For instance, the red cluster is composed by cells containing a number of grapevines ranging from 2 to 15 and at least one street which extension ranges from 11 to 1023 meters and adjacent to a growing in an area which extent ranges in [262..249975].

Other than exploring the behaviour of our system on the geomorphological clustering tasks, another goal of this experiment is to study, by maintaining the execution parameters specified above, if and how clustering results are affected by the chosen homogeneity threshold value or by some dataset properties such as data dimensionality (that is, the number of attributes describing each spatial object) and dimension of dataset (that is, the number of spatial objects to be clustered).

Concerning the homogeneity threshold, three additional runs have been performed employing values different from that one used above (0.99). Figure 5.15 depicts the resulting clusters obtained by ranging the homogeneity threshold values in $\{0.85, 0.90, 0.95, 0.99\}$.



(a) threshold=0.85 (b) threshold=0.90 (c) threshold=0.95 (d) threshold=0.99

Figure 5.15: Results on Ofanto dataset obtained with CORSO by using different homogeneity threshold values.

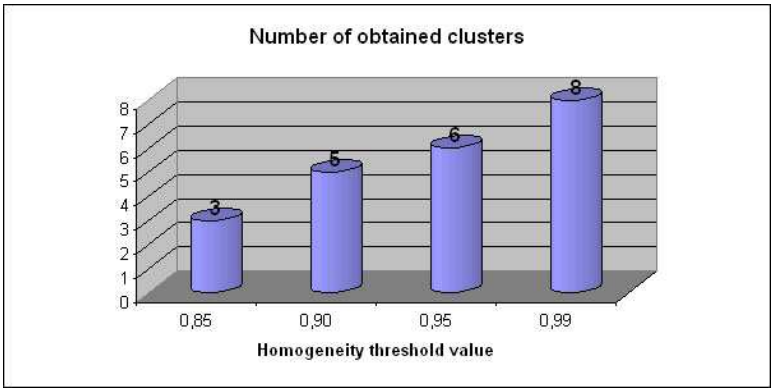


FIGURE 5.16: Ofanto dataset: number of clusters obtained by varying the homogeneity threshold value

As we expected, the number of clusters raises as the homogeneity threshold value increases (see Figure 5.16) since higher values of homogeneity threshold partition the space with higher levels of details. This happens because the higher is the homogeneity threshold, the more difficult it is for a group of spatial objects to be considered homogeneous and, hence, to be grouped together. This aspect can be easily observed by considering the red cluster of Figure 5.15.a (namely, the clustering task performed with $threshold = 0.85$), which homogeneity value is equal to 0.881: here, the set composed by the objects located from the third row to the end is considered to be homogeneous in the case (a), because its homogeneity estimation value (0.881) is greater than the fixed threshold (0.85), but it stops to be a cluster as soon as the homogeneity threshold exceeds such value.

Moreover, the above observation means that, in some sense, with high values of homogeneity threshold spatial objects are grouped by considering descriptors with a higher level of details and, hence, we expect that the cluster models tend to be as more accurate as the constraints of homogeneity are high. This hypothesis is confirmed by the comparison of cluster models obtained with different runs of the system. For instance, by focussing on the red cluster of Figure 5.15, the models M obtained by varying the homogeneity threshold values are:

$threshold = 0.85, 0.90$:

$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$
$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$
$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$

$\vee$

$belongs\_to\_cluster(X1) = true \leftarrow contains\_street(X1, X2) = true,$
$extension\_street(X2) \in [21..1228], street\_to\_parcel(X2, X3) = adjacent,$
$area\_growing(X3) \in [937..223300].$

$threshold = 0.95$:

$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$
$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$
$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$

$\vee$

$belongs\_to\_cluster(X1) = true \leftarrow contains\_street(X1, X2) = true,$
$extension\_street(X2) \in [36..1153], street\_to\_parcel(X2, X3) = adjacent,$
$area\_growing(X3) \in [1518..193687].$

$threshold = 0.99$:
$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$
$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$
$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$

Reader can observe that $M_{0.85} \prec M_{0.90} \prec M_{0.95} \prec M_{0.99}$ and, hence, the models become more and more restrictive as the homogeneity threshold value increases.

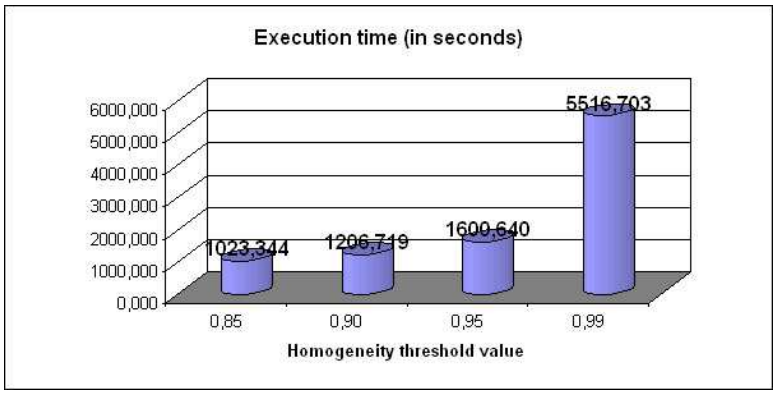


Figure 5.17: Ofanto dataset: execution time obtained by varying the homogeneity threshold value

Furthermore, Figure 5.17 shows that the running times (executions have been performed on a Pentium IV 3.2 GHz, 1 GB of RAM) raise roughly exponentially as the homogeneity threshold values increase. This is motivated by considering that most of execution time is spent to induce neighborhood models and this operation is more frequent for high values of homogeneity thresholds, where large clusters are rare. Results of application of CORSO by varying the homogeneity threshold values are summarized in Table 5.2.

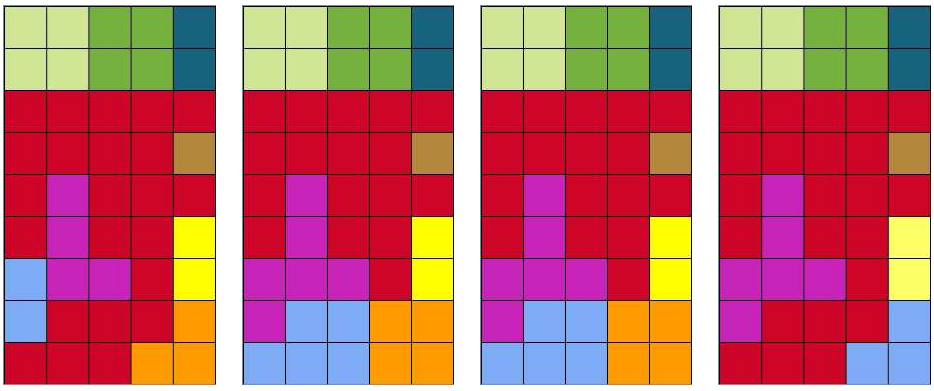
Threshold	**# Clusters**	**Avg objs per cluster**	**Exec. time (secs)**
0.85	3	15	1023.344
0.90	5	9	1206.719
0.95	6	7.5	1600.64
0.99	8	5.625	5516.703

Table 5.2: Ofanto dataset: CORSO runs with different homogeneity threshold values

Considering the data dimensionality, that is the number of attributes describing the considered spatial objects, different runs of CORSO have been performed by varying the number of descriptors (*descrs*) considered in the clustering tasks. Descriptors reduction has been performed by progressively removing the descriptors deemed of minor interest with respect to the experimental aims (see Table 5.3).

Descriptor Name	# considered descriptors			
	15	**20**	**25**	**31**
parcel_to_parcel	-	Y	Y	Y
canal_to_canal	-	-	Y	Y
canal_to_street	-	Y	Y	Y
canal_to_river	-	Y	Y	Y
street_to_street	-	-	Y	Y
street_to_river	-	Y	Y	Y
river_to_river	-	Y	Y	Y
canal_to_parcel	-	-	-	Y
street_to_parcel	-	-	-	Y
river_to_parcel	-	-	-	Y
railway_to_parcel	-	-	-	Y
bridge_to_parcel	-	-	-	Y
grapevines_to_parcel	-	-	Y	Y
olives_to_parcel	-	-	Y	Y
almonds_to_parcel	-	-	Y	Y
font_to_parcel	-	-	-	Y
contains_parcel	Y	Y	Y	Y
contains_canal	Y	Y	Y	Y
contains_street	Y	Y	Y	Y
contains_growing	Y	Y	Y	Y
contains_cultivation	Y	Y	Y	Y
contains_font	Y	Y	Y	Y
contains_river	Y	Y	Y	Y
contains_grapevines	Y	Y	Y	Y
contains_olives	Y	Y	Y	Y
contains_almonds	Y	Y	Y	Y
grapevines_in_cell	Y	Y	Y	Y
olives_in_cell	Y	Y	Y	Y
almonds_in_cell	Y	Y	Y	Y
area_growing	Y	Y	Y	Y
extension_street	Y	Y	Y	Y

Table 5.3: Ofanto dataset: descriptors considered in the data dimensionality variation tests



(a) descrs=15 (b) descrs=20 (c) descrs=25 (d) descrs=31

Figure 5.18: Results on Ofanto dataset obtained with CORSO by varying the data dimensionality.

Figure 5.18 shows that both the number and the shapes of the obtained clusters are very similar and seem to be unrelated with the descriptors used to induce cluster models. This surprising result can be explained on one hand by the criterion adopted in the descriptors reduction that removes the less important descriptors, and on the other hand by the presence of one or more alternative characterizations of the same cluster. This also explains the slight differences in the models associated with the same cluster. For instance, the red clusters of Figure 5.18, that has its upper part common to each run, is described by the following models:

number of considered descriptors = 15:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$$
$$contains\_street(X1, X3) = true, extension\_street(X3) \in [11..1023].$$

number of considered descriptors = 20:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$$
$$contains\_growing(X1, X2) = true, area\_growing(X2) \in [262..249975],$$
$$parcel\_to\_parcel(X3, X2) = meet, area\_growing(X3) \in [781..249975],$$
$$parcel\_to\_parcel(X4, X2) = meet.$$

number of considered descriptors = 25:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$$
$$contains\_growing(X1, X2) = true, area\_growing(X2) \in [262..249975],$$
$$parcel\_to\_parcel(X3, X2) = meet, area\_growing(X3) \in [781..249975].$$

number of considered descriptors = 31:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [2..15],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1023],$$
$$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$$

The above models show that the red cluster is characterized by cells containing

a number of grapevines ranging from 2 to 15 and two or more streets which length is enclosed in the interval [11..1023], one of them adjacent to a growing area (which extent is in [262..249975]). In addition, the models obtained with a number of descriptors equals to 20 or 25 reveal that the common part of the cluster also has at least two growing areas which maximum extent is 249975.

Analogous reasoning is valid for the other clusters.

This example suggests that different runs, each employing a different number of descriptors, can be profitably used to mine additional information about the calculated clusters (for instance, revealing properties characterizing sub-parts of the clusters). Finally, no regularity has been noticed in the execution times of the different runs (see Figure 5.19).

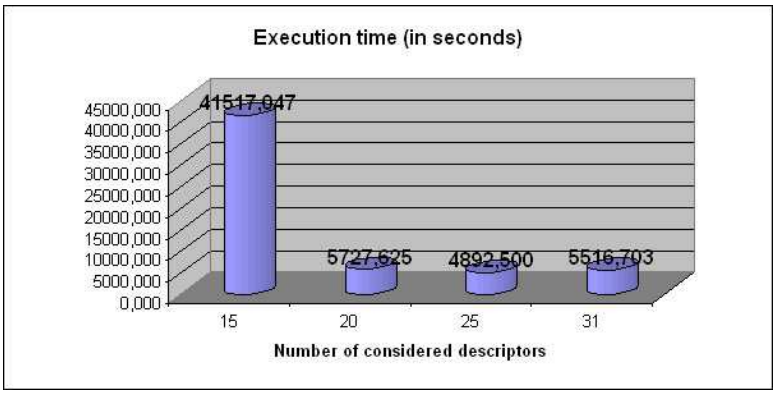
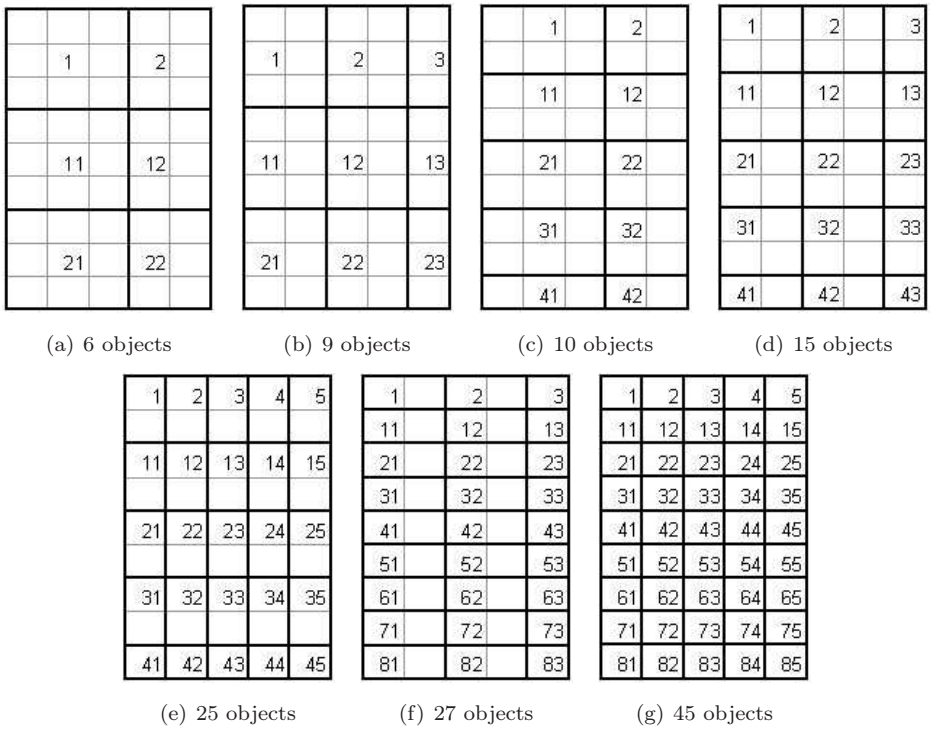


FIGURE 5.19: Ofanto dataset: execution time of different runs of CORSO obtained by varying the data dimensionality.

Considering the dimension of the dataset, that is the number of spatial objects to be clustered, CORSO has been applied to different partitions of the same territory. Here, the purpose is to vary the number of objects composing the dataset in order to investigate if it can be considered responsible of some modification in either the number or the models of the induced clusters. Variation of the dataset dimension is obtained by grouping in some way adjacent cells that, in turn, results in the analysis of the considered territory with different levels of granularity. The set of resulting datasets is reported in Figure 5.20.

In this figure, the basic grid in which the whole territory is partitioned is grey drawn in order to show how cells (that are black drawn) of each dataset are composed, while the number reported in each cell represents its identifier.

The application of CORSO to the datasets reported in Figure 5.20 results in a number of clusters that rises almost linearly with the increase of the considered objects as witnessed by Figure 5.21. This direct correlation is motivated by the greater easiness from wide cells to be merged together in that they tend to easily enclose common phenomena or elements. In other words, since a particular cultivation or a street with some specific features are more likely to occur in nearby locations (according to the *first Law of Geography*, [Tob79]), when such elements are located on the boundary of a cell they cause that cell to be clustered with one of its neighbors containing a similar element. Actually, in geomorphological characterization



(a) 6 objects (b) 9 objects (c) 10 objects (d) 15 objects

(e) 25 objects (f) 27 objects (g) 45 objects

Figure 5.20: Ofanto dataset: variation of the dimension of the dataset by means of different partitions of the territory.

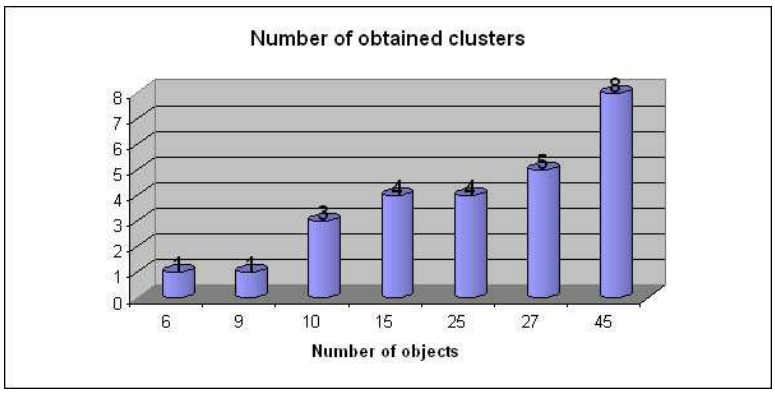
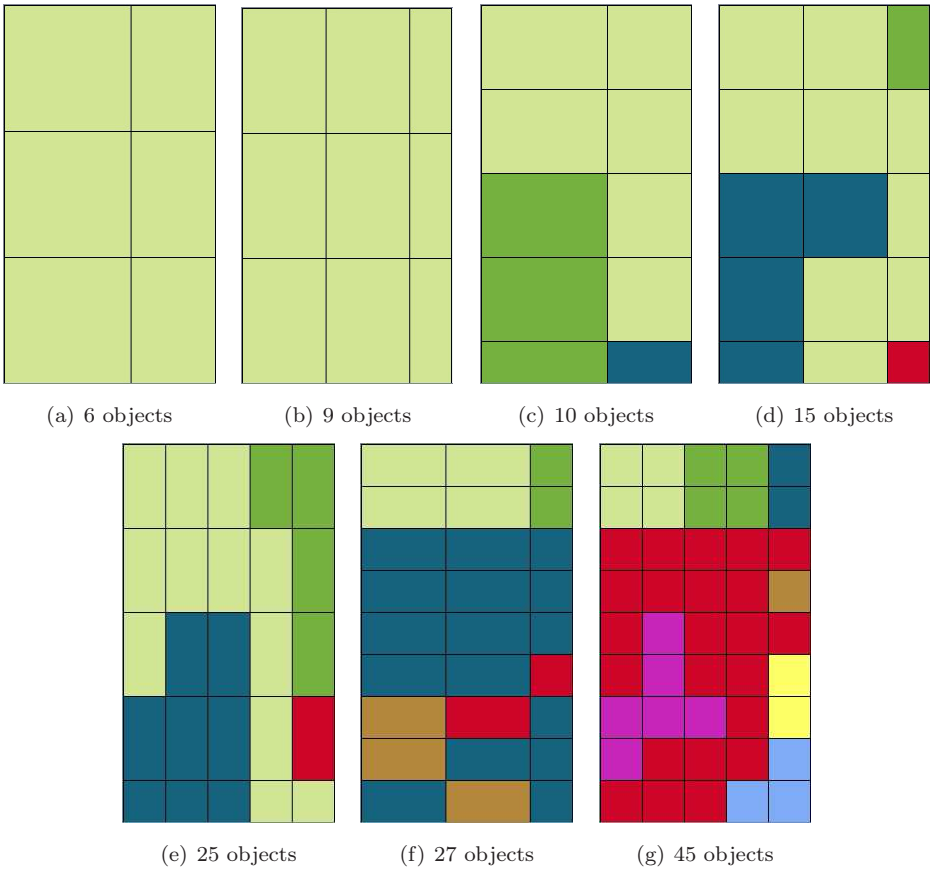


Figure 5.21: Ofanto dataset: clusters obtained by varying the dimension of the dataset



(a) 6 objects (b) 9 objects (c) 10 objects (d) 15 objects

(e) 25 objects (f) 27 objects (g) 45 objects

Figure 5.22: Clusters obtained by varying the dimension of the Ofanto dataset.

of territory it often happens that agricultural elements are present over the whole space but with different densities (e.g., olive trees in Mediterranean lands). As a consequence, the wider are the considered cells, the more likely is for two adjacent cells to contain some elements that can be considered similar to some others in an adjacent cell and, hence, to be merged together.

Figure 5.22 shows that different granularity of the considered cells leads to completely different clusters partitioning the same territory. This is basically due to the different models induced in each run. For instance, by looking at the pale green cluster (the first one induced), the models describing the cluster in each data configuration are:

6 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [37..124],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1228].$$

9 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow contains\_street(X1, X2) = true,$$
$$extension\_street(X2) \in [11..1179], street\_to\_parcel(X2, X3) = adjacent.$$

10 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [51..83],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1179].$$

15 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [33..67],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1179],$$
$$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..420112].$$

25 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow grapevines\_in\_cell(X1) \in [14..29],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1179],$$
$$street\_to\_parcel(X2, X3) = adjacent, area\_growing(X3) \in [262..249975].$$

27 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [2..7],$$
$$olives\_in\_cell(X1) \in [2..6], grapevines\_in\_cell(X1) \in [25..36],$$
$$contains\_street(X1, X2) = true, extension\_street(X2) \in [11..1179].$$

45 objects:

$$belongs\_to\_cluster(X1) = true \leftarrow almonds\_in\_cell(X1) \in [1..1],$$
$$contains\_font(X1, X2) = true, grapevines\_in\_cell(X1) \in [10..19].$$

It can be easily observed that, in general, the cluster is described in term of the number of grapevines and the extension of streets. The ranging intervals of these values, however, tend to raise as the extent of considered cells increases due to the

higher number of such elements in the cells. Differences in both the descriptors involved in the models and the intervals where they range are responsible of the number and the shapes of the resulting clusters. Again, no regularity has been noticed in the execution times of the above mentioned runs (see Figure 5.23).

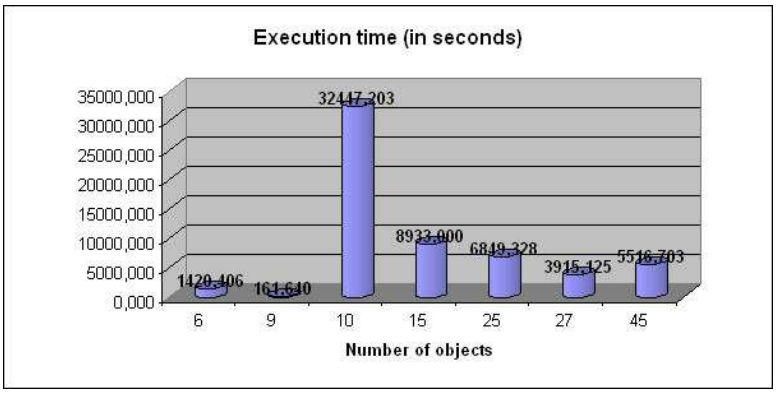


FIGURE 5.23: Ofanto dataset: execution time of different runs of CORSO obtained by varying the number of considered objects

5.4 Clustering geographical territories in North-West England relying on census data

In this application, we consider both census and digital map data concerning North West England (NWE) area, decomposed into censual sections (or *wards*) for a total of 1011 wards, with the aim of searching for possible relations between census status and geographical features.

The UK Census, undertaken every ten years, collects population and other statistics essential to those who have to plan and allocate resources. Census data provide aggregated information on demographic attributes such as person per household, cars per household, unemployment, migration, long-term illness.

Deprivation indices are selected as target variables, i.e. the analysis goal is to gain some information on attributive and spatial dependencies of these variables and their interactions. Information from the Census (sometimes in combination with other variables) is often combined into a single index score to show the level of deprivation in an area. Over the years a number of different such indices have been developed for different applications. In general, these measures show a strong correlation between the level of deprivation and a variety of health indicators.

In order to measure the deprivation level in the considered territory, we consider four deprivation index collected by 1998 Census: (i) *Jarman Underprivileged Area Score* (Jarman) that is designed to measure the need for primary care, the indices developed by (ii) *Townsend* and (iii) *Carstairs* that are used in health-related analysis, and the (iv) *Department of the Environment's Index* (DoE) that is used in targeting urban regeneration funds. Higher values of indices occur into more deprived wards.

The correlation of such indices and the Census variables is resumed in Table 5.4.

Variable	**Jarman**	**Townsend**	**Carstairs**	**DoE**
Unemployment	X	X	X (males)	X
Low social class	X		X	
Overcrowded households	X	X	X	X
Households lacking basic amenities				X
Single parent	X			
Under age 5	X			
Lone pensioner	X			
Residents who have changed address in the previous year	X			
Head of household born in the new commonwealth	X			
Households with no car		X	X	X
Not owner occupied		X		
Children living in flats				X
Children in low earning households				X
Low educational participation				X
Low educational attainment				X
Standard Mortality Ratios				X
Male long term unemployment				X
Income Support recipients				X
Home Insurance Weightings				X

Table 5.4: Census data analysis: variables used in the calculation of four deprivation indices

Spatial analysis of deprivation distribution is enabled by the availability of vectorized boundaries of the 1011 census wards as well as by other Ordnance Survey digital maps of NWE, where several interesting layers are found, namely urban zones (including 384 large urban areas and 2232 small urban areas) and wood zones (including 859 woods). In particular, we focus our attention on investigating continuity of socio-economic deprivation joined to geographical factors represented in linked topographic maps.

Both ward-referenced census data and map data are stored in an Object-Relational spatial database, i.e., Oracle Spatial 9i database, as a set of spatial tables, one for each layer. Each spatial table includes a geometry attribute that allows storing the geometrical representation (i.e. urban and wood zones are described by lines while wards are described by polygons) and the positioning of a spatial object with

respect to some reference system. We adopt a topological algorithm based on the 9-intersection model [Ege91] to detect both adjacency relation between NWE wards (i.e. wards which share some boundary) and overlapping relation between wards and urban areas (or woods). The former imposes a discrete spatial structure over NWE wards such that only adjacent wards may be grouped in the same cluster while the latter contributes to define the spatial structure embedded in each ward not only in terms of observed values of deprivation scores but also extension of urban areas and/or woods overlapping each ward. No BK is defined for this problem.

As already observed in the Ofanto experiment described in Section 5.3, the granularity of partitioning changes when varying the value of homogeneity threshold ($h-threshold$), again leading to a direct correlation between the value of the homogeneity threshold and the number of obtained clusters. In this case, CORSO detects 79 clusters with $h-threshold = 0.80$, 89 clusters with $h-threshold = 0.85$, 122 clusters with $h-threshold = 0.90$ and 163 clusters with $h-threshold = 0.95$. In particular, when $h-threshold = 0.95$, CORSO clusters NWE area in 2160 secs (executions have been performed on a Pentium IV 3.2 GHz, 1 GB of RAM) and identifies adjacent regions modeling different relational patterns involving deprivation and geographical environment.

For instance, by analyzing these spatial clusters, we discover three adjacent areas, namely C_1, C_2 and C_3 compactly labeled as follows:

$C_1 : cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-4.7..-0.6],$
$doe(X_1) = [-12.4..2.7], carstairs(X_1) = [-4.5..-0.9],$
$jarman(X_1) = [-32.7..7.5], overlapped\_by\_wood(X_1, X_2) = true.$
$cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-5.4..-2.3],$
$doe(X_1) = [-10.9..-0.5], carstairs(X_1) = [-4.2..-1.6],$
$jarman(X_1) = [-22.8..0.6], overlapped\_by\_wood(X_1, X_2) = true.$
$cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-5.4..-3.2],$
$doe(X_1) = [-8.8..-2.1], carstairs(X_1) = [-4.4..-2.5],$
$jarman(X_1) = [-22.8..-2.4], overlapped\_by\_wood(X_1, X_2) = true.$

$C_2 : cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-2.0..0.6],$
$doe(X_1) = [-4.2..1.6], carstairs(X_1) = [-2.6..2.1],$
$jarman(X_1) = [-9.7..8.8], overlapped\_by\_largeUrbArea(X_1, X_2) = true.$
$cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-2.7..2.8],$
$doe(X_1) = [-4.2..4.0], carstairs(X_1) = [-2.2..2.7],$
$jarman(X_1) = [-8.8..21.3], overlapped\_by\_largeUrbArea(X_1, X_2) = true$

$C_3 : cluster(X_1) = c_1 \leftarrow townsend(X_1) = [-3.4..0.4],$
$doe(X_1) = [-8.2..-0.2], carstairs(X_1) = [-3.7..0.6],$
$jarman(X_1) = [-27.7..-1.5],$
$overlapped\_by\_smallUrbArea(X_1, X_2) = true.$

C_1, C_2 and C_3 cover adjacent areas with quite similar range values for deprivation indices but C_1 models the presence of woods while C_2 and C_3 model the presence of small urban areas and large urban areas, respectively. In particular, by observing the intervals in which deprivation indices range, it can be observed that areas containing large urban conglomerations (cluster C_2) are characterized by quite high values of

Carstairs and *Townsend* indices, while both the small urban areas (cluster C_3) and that ones enclosing forestal territories (cluster C_1) have lower values for *Jarman* and *DoE* indices. This surprisingly demonstrates that, perhaps due to the high density of population and the presence of suburbs surrounding large cities, large urban areas suffer from problems related to health and economic deprivations in population.

Discontinuity of geographical environments modelled by these clusters is confirmed by visualizing map data about the area (see Figure 5.24).

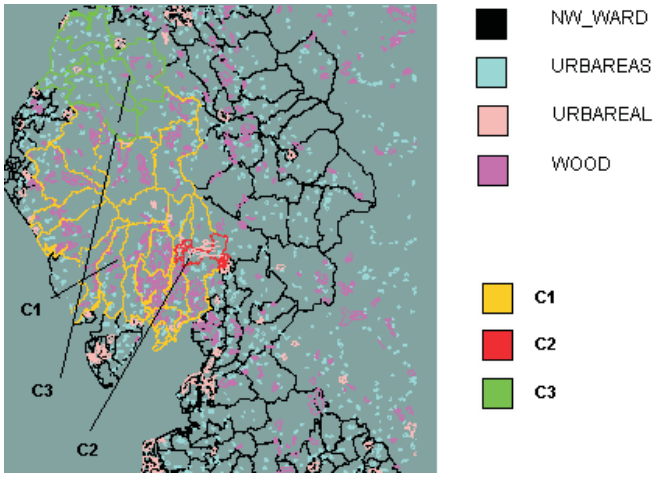


Figure 5.24: Spatial clusters detected on NWE with homogeneity threshold = 0.95.

5.5 Mining and characterizing the distribution of air pollution in Apulia

This section describes the application of CORSO to a real-world domain that has been considered of primary importance in last decades and is undoubtedly candidate to become one of the most funded research fields before long: the pollution analysis and monitoring.

The application of Data Mining techniques to this domain is an attractive but challenging task, since advances of such a discipline could be profitably used to aid humans to solve some serious problems affecting earth. However, properly gathering and handling such data and mining information really useful for biologists, geologists and environmental scientists is particularly hard since data are often heterogeneous, missing or noisy. In addition, available data are often meaningless in raw form they are collected and, hence, they are hard to be interpreted without the knowledge owned by an expert.

The experiment described in this section focusses on air pollution analysis concerning the Apulia area (Italy) in 2005 and aims at applying CORSO with a twofold purpose: (i) finding geographical regions of Apulia affected by similar air pollution problems and (ii) generating logical models summarizing the peculiarities of each group.

Environmental data have been provided by *ARPA Puglia* agency (Agenzia Regionale per la Protezione Ambientale, *http://www.arpa.puglia.it*), a regional state agency whose main purpose is to measure and monitor the level of pollution (in all its possible facets, such as air, water, soil and acoustic) in the territory of Apulia. They consist in lists of surveys hourly gathered from 01/01/2005 (00:00) to 31/12/2005 (23:00) by both fixed and mobile testing stations specifically designed to measure air pollution.

Even though the kind collaboration of ARPA Puglia in supplying data gathered through 2005, analysis of such data is complicated by the following three factors:

- data are gathered by three different and not related monitoring networks (RRQA, SIMAGE and Provincial network) and concern five districts, with 36 Air Pollution Testing Stations (*APTSs*) in total (see Table 5.5)
- data gathered by the APTSs differ in the form they are stored (an Excel file for the stations encompassed in Bari, Foggia and Lecce districts with as many sheets as the number of considered stations and CSV files for the stations belonging to Taranto and Brindisi districts) and in pollutants observed (each APTS is able to measure the concentration in the air of a restricted set of pollutants)
- APTSs are irregularly arranged, so it is impossible to have a precise and continuous snapshot of the pollution situation in Apulia

In order to overcome the difficulties concerning the first and the second items of the list above and to transform data from the available format into another one suited to be read by CORSO, a two-step procedure has been performed:

District	**RRQA Network**	**SIMAGE Network**	**Provincial Netw.**	#
Bari	via Caldarola (BA) Ciapi (BA) Ex-Enaip (Modugno) Z.I. ASM (Molfetta) P.za Verdi (Molfetta)			**5**
Brindisi	Mesagne Torchiarolo S. Pietro Vern. S. Pancrazio S. via Taranto (BR)	P.za S. Giusto (BR) Bozzano via dei Mille (BR) SISRI		**9**
Foggia	Cap. Porto (Manfred.) v. Mandorli (Manfred.) v. Michelangelo (Manfr.) sc. Ungaretti (Manfred.) Ciuffreda (M. S. Angelo)			**5**
Lecce	S. M. Cerrate (LE) Giorgilorio (Surbo) Baldassarri (Guagnano) zona Riesci (Arnesano) S. Barbara (Galatina)			**5**
Taranto	v. Archimede (Tamburi) S. Vito v. Adige (TA) v.Macchiavelli (Tamburi) v. Sorgenti (Statte)	v. Foscolo (Talsano) v. Speziale (TA) CISI (q.re Paolo VI) s.s.7 - Wind (Statte)	Grottaglie Martina Franca Manduria	**12**
Total	**25**	**8**	**3**	**36**

Table 5.5: ARPA dataset: arrangement of the Air Pollution Testing Stations in the Apulia districts

1. the Excel file has been converted into a number of CSV files, in order to have a unique format for all the APTSs considered in this experiment;

2. a Java application has been deliberately developed with the aim of generating the CORSO dataset from the CSV files.

The result of this transformation is a XML file enclosing 36 objects describing the considered APTS. The structure of each object is depicted in Figure 5.25

FIGURE 5.25: The structure of a CORSO object describing an Air Pollution Testing Station

The body of the example reported in Figure 5.25 can be considered as roughly structured into four parts:

- the *Symbolic literals* section, that is the set of literals defining the general features of the station (the district, the municipality and the network it belongs to, the type of the station and the type of the zone it is placed, etc.)

- the *Month sequence* section, a set of literals expressing the relations between months involved in the experiment

- the *Numerical literals* section, enclosing all the literals describing the minimum, maximum and average values of surveys grouped by month

- the *Aggregate literals* section, enclosing all the literals describing the minimum, maximum and average values of surveys calculated over the entire year

As concerns the considered pollutants, in order to focus on main aspects of air quality, we have only taken into account information concerning the observed pollutants leaving out information about the atmospheric conditions (pressure, humidity,

temperature, wind velocity and direction). One could claim that atmospheric information could affect the pollution levels of surrounding lands. However, the purpose of this experiment is to detect areas characterized by similar pollution phenomena without investigating on their possible causes.
The complete list of considered pollutants is: CO, SO_2, NO_x, NO, NO_2, Pts, PM_{10}, $Benzene$, $EBenzene$, H_2S, O_3, $Toluene$, $Xyleni$, $M-Pxylene$, $O-Xylene$, $Ch4$, $Nmhc$.

The unit of measurement concerning the above pollutants is mg/m^3 for CO and $\mu g/m^3$ for the other ones. Details on each pollutant considered in this experiment are reported in [LANP03], while pollutant limits fixed by Italian law can be found in $D.M.60/02$ and $D.LGS.183/04$.

As stated above, each APTS is able to only measure a restricted set of pollutants according to the area to monitor (urban, suburban or rural) and the purpose of the survey (traffic, industrial or generic). Details on each APTS are reported in Tables 5.7 and 5.8, while Figure 5.26 shows the geographical arrangement of APTSs belonging to each district. From this figure it is easy to see how the arrangement of APTSs is uneven both in the number of stations per district and in the spatial distribution.

Till now we have considered APTSs as isolated objects without defining relations among them. In order to define the discrete spatial structure over the data, the following relation has been considered:

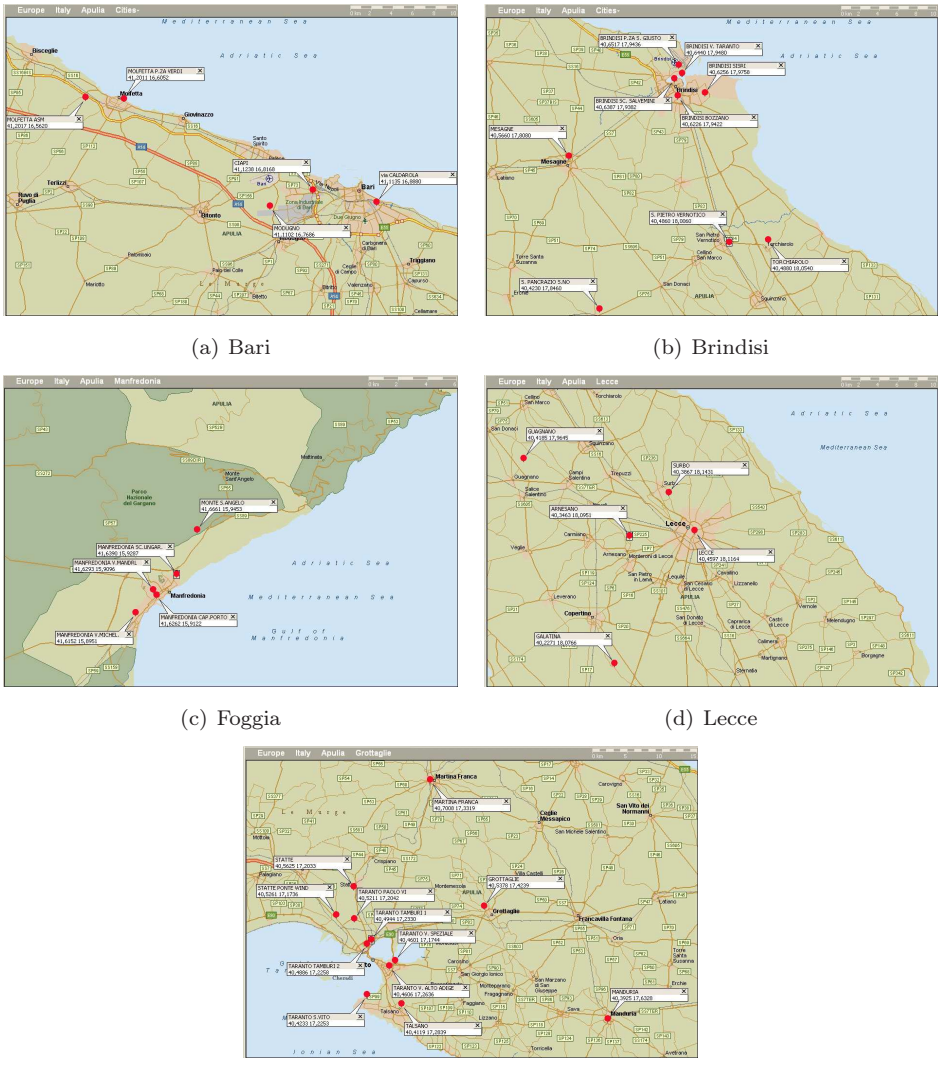
"*Two objects (Air Pollution Testing Stations) are related each other iff there is at least one main road linking them*"

This relation leads to link together testing stations that result to be directly reachable each other and this roughly corresponds to link a testing station to that ones that are geographically closest. Figure 5.27 shows two partial views of such discrete spatial structure relative to the part of Apulia encompassing Foggia and Bari (Fig. 5.27.a) and Taranto, Brindisi and Lecce (Fig. 5.27.b) districts, respectively, while Figure 5.28 shows the comprehensive view of APTSs in Apulia and their links.

CORSO has been applied to the above described dataset with the aim of checking for regularities in pollution of observed areas. Application has been performed by using the following clustering parameters:

- $threshold = 0.95$
- $seedselection = doc$ (Descending Order of Connectivity)
- $punctualAdding = true$

These choices are motivated by the need for a clustering task that is able to discover groups of stations gathering quite similar pollutant values ($threshold = 0.95$), that starts to search clusters from the most central (or connected) stations (descending order of connectivity criterion for the seed selection) and that checks for



(a) Bari

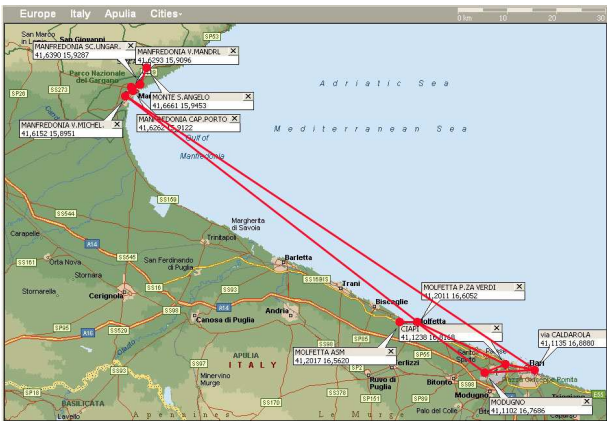
(b) Brindisi

(c) Foggia

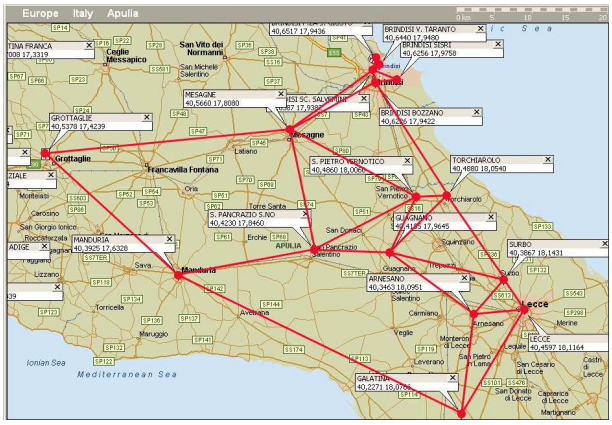
(d) Lecce

(e) Taranto

Figure 5.26: Geographical arrangement of Air Pollution Testing Stations belonging to the five Apulian districts



(a) Foggia and Bari linked testing stations



(b) Taranto, Brindisi and Lecce linked testing stations

Figure 5.27: Partial views of discrete spatial structure defined over the Apulian Air Pollution Testing Stations

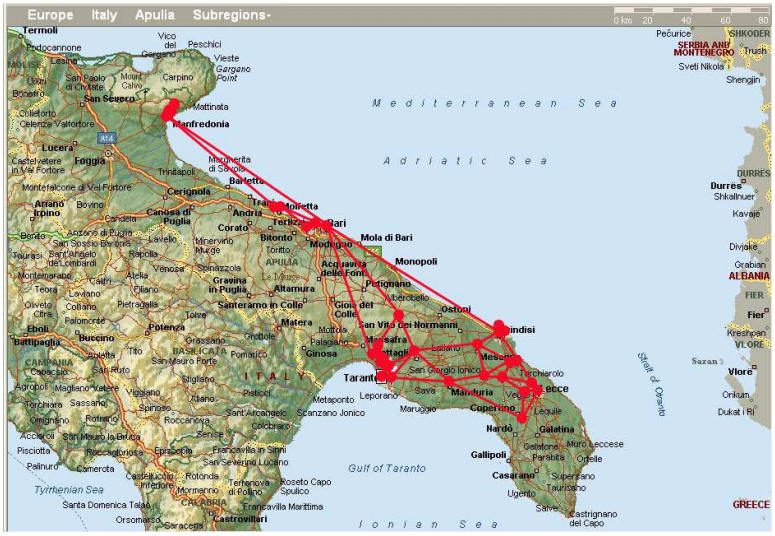


Figure 5.28: Comprehensive view of the Air Pollution Testing Stations in Apulia and their links

the single-neighbor cluster expansion whenever a neighborhood cluster expansion fails (*punctualAdding*).

As concerns the ATRE learning engine settings (that is, the parameters that guide the induction of cluster models), the goal is to obtain models composed by as detailed as possible clauses (in terms of number of composing literals) since the higher is the number of literals involved in the intensional description of a cluster, the more is the information extracted by that cluster. To this aim, we have defined the preference criteria for the clause selection by giving primary importance to the maximization of the number of literals with respect to the number of covered examples. The resulting preference criteria for the clauses selection are the following:

$best\_lef([[numlet, max, 0.0], [cost, min, 0.0], [poscov, max, 0.0]])$.

$ps\_lef([[numlet, max, 0.0], [cost, min, 0.0], [poscov, max, 0.0]])$.

In addition, due to the high number of descriptors involved in each learning task, preferences on the descriptors to add into models have been defined by means of weights. In particular, a greater importance has been assigned to descriptors devoted to represent maximum and average values ($weight = -1$) to the detriment of descriptors devoted to represent minimum values ($weight = 1$). This choice is motivated by the assumption that maximum and average values of pollutants are typically more indicative of some phenomenon occurrence than the minimum values. For instance, in a traffic area scientists typically pay more attention to maximum (or average) values of benzene with respect to minimum peaks.

The clustering task performed by CORSO generates eight clusters while four objects are labelled as noise since they are not assigned to any cluster (see Figure 5.29 for a graphical representation of clusters arrangement and Table 5.6 for a summarizing view of detected clusters).

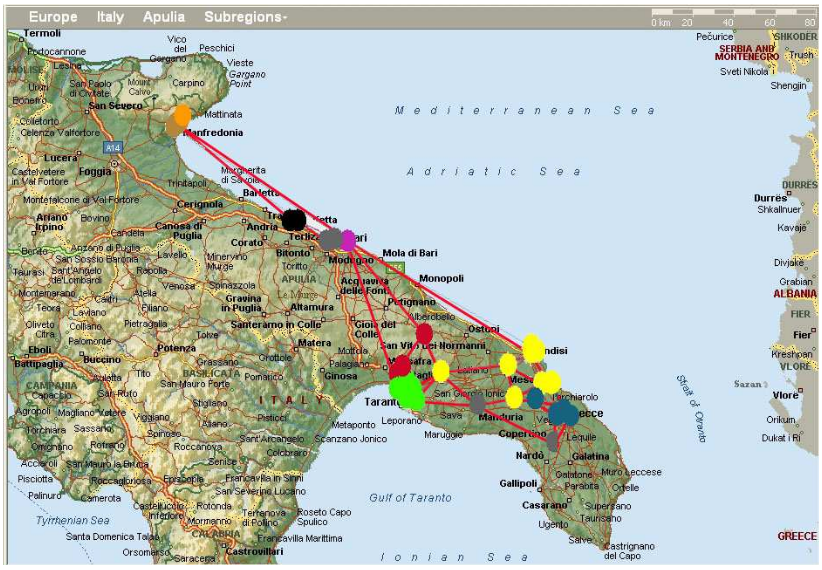


Figure 5.29: ARPA dataset: results of clustering task performed by CORSO

Cluster ID	**Assigned color**	**Homogeneity value**	**Card.**	**APTS**
C1	yellow	0.9781190281	9	Mesagne, Bozzano, via dei Mille, S. Pietro Vern., S. Pancrazio S., Grottaglie, SISRI, Torchiarolo, P.za S. Giusto
C2	green	1.0	6	v. Adige, v. Speziale, S. Vito, Talsano, v. Archimede, v. Machiavelli
C3	blue	1.0	4	Giorgilorio, zona Riesci, Baldassarri, S. M. Cerrate
C4	red	1.0	4	CISI, v. Sorgenti, Martina Franca, s.s.7 - Wind
C5	brown	1.0	4	Cap. di Porto, v. Mandorli, v. Michelangelo, sc. Ungaretti
C6	fuchsia	1.0	2	via Taranto, via Caldarola
C7	black	1.0	2	Z.I. ASM, P.za Verdi
C8	orange	1.0	1	Ciuffreda
NOISE	gray	*not available*	4	Manduria, Ciapi, Ex-Enaip, fraz. S. Barbara

Table 5.6: ARPA dataset: arrangement of the Air Pollution Testing Stations in the Apulia districts

Table 5.6 let us to remark some observations:

- the third column reports the homogeneity values of detected clusters; such values can be considered as an indicator of quality for the performed clustering task since they express the "purity" level of each detected cluster
- the number of objects per cluster decreases while moving from the clusters detected in the early stages of the task towards the last detected ones; this indicates that CORSO tends to detect the largest clusters first when adopting the "*Ascending Order of Connectivity*" as the seed selection criterion
- $C8$ is composed by only one object; this object is considered to be a cluster instead of being labelled as noise because it is selected as seed object when all their neighbors have already been assigned to clusters that are not homogeneous with respect to it. In this case, the neighborhood of this object is only composed by the object itself and, hence, is obviously homogeneous.

As concerns the models of detected clusters, they actually capture the common features of the objects belonging to each cluster. Examples of cluster models are reported below:

$C4 : cluster(X_1) = c_4 \leftarrow nox\_avg\_value(X_1) = [18.7701..43.9805],$
$nox\_max\_value(X_1) = [264.8772..889.2056],$
$no\_avg\_value(X_1) = [3.16..12.7115],$
$no\_max\_value(X_1) = [108.3157..461.5783],$
$no2\_avg\_value(X_1) = [14.1028..24.8345].$
$C5 : cluster(X_1) = c_5 \leftarrow so2\_avg\_value(X_1) = [0.7333..3.0569],$
$so2\_max\_value(X_1) = [6.6516..93.0218],$
$no2\_avg\_value(X_1) = [18.6143..42.6376],$
$no2\_max\_value(X_1) = [115.7727..233.0003],$
$district(X_1) = fg, network(X_1) = rrqa,$
$municipality(X_1) = manfredonia, zonetype(X_1) = suburban.$

The above clauses describe the clusters $C4$ and $C5$ in terms of their distinctive features. For instance, the first clause states that cluster $C4$ is composed by areas with similar average values for NO, NO_X and NO_2 and similar maximum values for NO and NO_X, whose ranges are reported in the clause. In particular, the second clause shows that in the case of cluster $C5$ CORSO has been able to generate a model expressing that the Air Pollution Testing Stations belonging to this cluster, other than being characterized by similar values of some pollutants, are placed in the same municipality, belong to the same network and work in a suburban zone. In other words, it has been able to state that the APTSs in the municipality of $Manfredonia$ have been considered similar not only because they are close each other and belong to the same area, but also because they are similar according to the descriptors measuring the average and maximum values for NO_2 and the maximum values for SO_2.

However, since in this dataset objects are almost completely composed by numeric literals, the quality of induced models could be further improved by enriching the model generation phase with clause selection criteria specifically suited for numeric descriptors. In particular, it would be useful to have a criterion that would prefer literals where the ratio of interval width of considered values out of the width of the descriptor domain is as small as possible.

For instance, let us assume that $Benzene$ values can range between 0 and 1000 while CO is defined in [0..50]. Let us consider the following clauses[1]:

$A : cluster(X) = true \leftarrow \ldots, benzene\_avg\_value(X) = [10..15], \ldots$
$(score : (15 - 10)/(1000 - 0) = 5/1000 = 0.005)$
$B : cluster(X) = true \leftarrow \ldots, benzene\_avg\_value(X) = [80..400], \ldots$
$(score : (400 - 80)/(1000 - 0) = 320/1000 = 0.32)$
$C : cluster(X) = true \leftarrow \ldots, co\_avg\_value(X) = [30..35], \ldots$
$(score : (35 - 30)/(50 - 0) = 5/50 = 0.1)$

In this case, the desired clause to prefer should be the first one (clause A) because it results to be the more restrictive taking into account the range of values that fire the clauses and the possible values of the pollutants. Actually, clause A seems to be that one that, more than the others, can capture peculiarities of the group of objects to cluster.

[1] in the parentheses are reported the scores of the clauses according to the proposed criterion

District	Network	Municipality	Address	Zone Type	Station Type	UTM 33 Coord.		Observed Pollutants
						East	North	
Bari	RRQA	Bari	via Caldarola	urban	traffic	658520	4553079	SO_2, NO_2, NO_X, CO, O_3, $Benz$, PM_{10}
		Bari	Ciapi	suburban	traffic / industrial	652513	4554095	SO_2, NO_2
		Modugno	Ex-Enaip	suburban	industrial	648498	4552501	SO_2, NO_2, NO_X, CO
		Molfetta	Z.I. ASM	suburban	generic	630969	4562323	SO_2, NO_2, NO_X, O_3
		Molfetta	P.za Verdi	urban	traffic	634595	4562323	SO_2, NO_2, NO_X, PM_{10}
Brindisi	RRQA	Mesagne	Mesagne	suburban	generic	737714	4494370	SO_2, NO_2, NO_X, NO, Pts
		Torchiarolo	Torchiarolo	suburban	generic	758842	4486404	PM_{10}, CO, SO_2, NO_2, NO_X, NO, Pts
		S. Pietro Vern.	S. Pietro Vern.	suburban	generic	754781	4486042	SO_2, NO_2, NO_X, NO, Pts
		S. Pancrazio S.	S. Pancrazio S.	suburban	generic	741444	4478597	PM_{10}, SO_2, NO_2, NO_X, NO, Pts
		Brindisi	via Taranto	urban	traffic	749277	4503418	CO, SO_2, NO_2, NO_X, NO, O_3, Pts, Ch_4, $NmHc$, $H2s$
	Simage	Brindisi	P.za S. Giusto	urban	industrial	748879	4504259	PM_{10}, SO_2, NO_2, NO_X, NO
		Brindisi	Bozzano	urban	industrial	748869	4501030	PM_{10}, SO_2, NO_2, NO_X, NO
		Brindisi	via dei Mille	urban	industrial / traffic	748464	4502807	PM_{10}, SO_2, NO_2, NO_X, NO
		Brindisi	SISRI	suburban	industrial	751700	4501449	PM_{10}, SO_2, NO_2, NO_X, NO, CO, $Benz$, Tol, $M-pxyl$, $O-xyl$
Foggia	RRQA	Manfredonia	Cap. di Porto	suburban	traffic	575991	4608679	SO_2, NO_2
		Manfredonia	v. Mandorli	suburban	traffic	575770	4609022	SO_2, NO_2, NO_X, CO, O_3, $Benz$, PM_{10}
		Manfredonia	v. Michelangelo	suburban	traffic	574576	4607442	SO_2, NO_2, CO
		Manfredonia	sc. Ungaretti	suburban	industrial	577344	4610110	SO_2, NO_2
		M. S. Angelo	Ciuffreda	rural	generic	578692	4613137	SO_2, NO_2

Table 5.7: ARPA dataset: details of the Air Pollution Testing Stations (Bari, Brindisi and Foggia districts)

District	Network	Municipality	Address	Zone Type	Station Type	UTM 33 Coord. East	UTM 33 Coord. North	Observed Pollutants
Lecce	RRQA	Lecce	S. M. Cerrate	rural	generic	764242	4483446	SO_2, NO_2, NO_X, CO, O_3, $Benz$, PM_{10}
		Surbo	Giorgilorio	suburban	generic	766797	4475426	SO_2, NO_2, NO_X, CO
		Guagnano	Baldassarri	suburban	traffic	751513	4478431	SO_2, NO_2, NO_X, CO, O_3, PM_{10}
		Arnesano	zona Riesci	suburban	traffic	762876	4470790	SO_2, NO_2, NO_X, CO, O_3, PM_{10}
		Galatina	fraz. S. Barbara	suburban	traffic	761767	4457503	SO_2, NO_2
Taranto	RRQA	Tamburi	v. Archimede	suburban	industrial	689238	4485033	SO_2, NO_2, NO_X, NO, Pts, $H2s$, CO, PM_{10}
		Taranto	S. Vito	suburban	traffic / industrial	688778	4477122	SO_2, NO_2, NO_X, NO, Pts
		Taranto	v. Adige	urban	traffic	691924	4481337	SO_2, NO_2, NO_X, NO, Pts
		Tamburi	v. Macchiavelli	suburban	industrial	688642	4484370	SO_2, NO_2, NO_X, NO, CO, O_3, $Benz$, PM_{10}
		Statte	v. Sorgenti	suburban	industrial	686530	4492525	SO_2, NO_2, NO_X, NO, PM_{10}, Pts
	Simage	Talsano	v. Foscolo	suburban	industrial	693783	4475985	SO_2, NO_2, NO_X, NO, PM_{10}
		Taranto	v. Speziale	rural	industrial	684358	4481091	SO_2, NO_2, NO_X, NO, PM_{10}
		q.re Paolo VI	CISI	rural	industrial	686716	4487932	SO_2, NO_2, NO_X, NO, PM_{10}
		Statte	s.s.7 - Wind	rural	traffic / industrial	684114	4488423	SO_2, NO_2, NO_X, NO, CO, PM_{10}, $Benz$
	provincial	Grottaglie	Grottaglie	suburban	generic	705279	4490271	SO_2, NO_2, NO_X, NO, CO, O_3
		Martina Fr.	Martina Franca	urban	traffic	697012	4508162	NO_2, NO_X, NO, CO, $Benz$, O_3, Pts
		Manduria	Manduria	urban	traffic	723453	4474650	NO_2, NO_X, NO, Pts, CO, $Benz$, O_3, Tol, Xyl

Table 5.8: ARPA dataset: details of the Air Pollution Testing Stations (Lecce and Taranto districts)

5.6 Conclusions

In this chapter, we have discussed five applications of the clustering method proposed in this thesis; two of them concern artificial data, while the other ones are about real-world domains.

The first experiment concerns the application of CORSO to a dataset describing computers composing a LAN. Here, each object represents a computer in term of a set of descriptive features (operative system installed, type of IP assignment, type of connection, etc.) while the links among objects are defined by the adjacency relation of the corresponding computers. The goal of this experiment is to test the method over data configurations that in literature are deemed to be particularly difficult to be handled by clustering methods. To this aim, data have been deliberately arranged to originate irregularly-shaped and nested clusters. Results show that, with a proper configuration of learning parameters, CORSO can discover both nested and irregularly-shaped clusters.

The second experiment describes the application of CORSO to the domain of Social Network Analysis. In particular, the goal is to group people working into a company according to their similarities (past experiences, professional ability, working projects, etc.) and their relations (email exchanged or phone calls, for instance). This experiment demonstrates that CORSO, and clustering in general, can be profitably used to suggest the arrangement of people according to the type of knowledge hold by the informal working teams.

The remaining applications concern geographical domains. Data concerns agricultural information of an Apulian region, social indicators and geomorphological features of the North-West England territory and air pollution surveys in Apulia, respectively. In particular, with reference to the first geographical application, CORSO has been applied with the aim of extracting territory characterization of lands in proximity of Canosa di Puglia. This experiment has been performed with the aim of studying if and how clustering results are affected by the chosen homogeneity threshold value or by some dataset property such as dataset dimension or data dimensionality. Results show that the number of obtained cluster grows almost linearly as either the threshold value or the number of objects composing the dataset increases, while runs have demonstrate that dimensionality of data (that is, the number of attributes employed in the object descriptions) does not affect the number and the shape of detected clusters.

Finally, comparisons of CORSO with other clustering algorithms (K-Means, Expectation Maximization, DBScan and Cobweb) have been performed in the LAN artificial application demonstrating that CORSO generally outperforms algorithms working only on spatial or conceptual features of data since it is able to take into account both of these aspects.

Chapter 6

Conclusions

In this dissertation we have presented a new Data Mining method that is able to cluster structured objects according to their structural similarity and that takes into account one or more binary (spatial) relations existing among them. We conclude with an overall summary of this thesis contribution and discuss a number of possibilities for future work.

6.1 Summary

Clustering is deservedly deemed one of most important and challenging data mining task: it is important because grouping objects into a meaningful way, independently from the adopted criteria, is considered to be symptom of intelligent behaviour; it is challenging because this task involves a series of difficulties that have to be taken into account. In this dissertation we have revised the current state of research carried out in clustering, analysing the main approaches developed and, for each of them, some well-known algorithms, their strengths and weaknesses.

In particular, as concerns the criteria adopted to cluster data, we have focussed our attention on graph-based partitioning algorithms and on conceptual clustering. Graph-based partitioning methods consider observations as nodes of a graph whose arcs indicates relations among observations. Here, clustering consists in partitioning the graph according to nodes connectivity or other graph structural characteristics. However, methods belonging to this approach basically focus on links among nodes (for instance considering strength of links or sub-graphs occurrences) paying poor attention to the similarity of observations. Such similarity is a key aspect for conceptual clustering that considers each cluster as a concept and partition training objects on the basis of the symbolic descriptions which can be associated with each cluster. Data are, however, considered to be entities independent each others, so this approach does not work well in many environments, where interactions among observed units exist and cannot be neglected.

As to the data, several clustering algorithms assume that observations are represented as independent tuples of a single database relation (single-table assumption). In reality, two complications makes this assumption unacceptable for real-world do-

mains: (i) data to consider are usually heterogeneous and, hence, require to be scattered over different tables of a database and (ii) relations exist among observations. This lead to conceive data as complex entities composed by a their own internal structure. (Multi-)Relational Data Mining approaches overcomes limitations of single-table assumption by typically resorting to ILP-based techniques. Nevertheless, MRDM generally works in learning from interpretation settings, where the basic assumption is independence of observations. This constraint is too restrictive in particular environments, such as spatial domains, where often attributes of an observation are somehow affected by the values assumed by some attribute of neighbors.

On the other hand, spatial clustering methods work by grouping objects on the basis of spatial attributes and relations as witnessed by different methods belonging to the approaches reported in this dissertation. Peculiarities and weaknesses of each of them are discussed, but the common limitation is that all spatial methods cluster data without caring about the resemblances resulting by their internal structure.

To overcome the limitations discussed above, we have proposed a multi-relational clustering method, named CORSO (Clustering Of Related Structured Objects), that is also able to take into account relations existing between objects in cluster detection. In particular, following a spatial clustering approach, it detects group of objects by repeatedly merging neighborhoods of objects. Such neighborhoods are conceived as group of objects that are linked according to some (spatial) binary relation and homogeneous each others. Briefly, the proposed method uses the graph-based partitioning methods formalism to represent data and relations, detects neighborhoods following the suggestions provided by a density-based clustering approach and evaluates homogeneity of sets of objects by associating them with logical models like in conceptual clustering approaches.

Applications to both artificial and real-world domains have been performed in order to validate the proposed method with data of different size, types and peculiarities. Artificial datasets have been used to test CORSO on data configurations that are known in literature to be difficult to cluster. Experimental results have shown that CORSO, differently form a lot of well-known spatial clustering methods in literature, is able to detect nested clusters and irregularly shaped ones. In addition, it has been shown that this method is also able to detect partially overlapping clusters, where the same observation can belong to two or more contiguous clusters. Moreover, referring to an artificial dataset, results obtained with CORSO have been compared with results obtained by using different clustering algorithms (K-Means, Expectation Maximization, DBScan and Cobweb). CORSO outperforms the considered clustering algorithms thanks to its ability of simultaneously taking into account both the spatial and the descriptive features of objects.

Since an important problem in clustering is the effect of dimension of data (number of objects to cluster) and data dimensionality (number of features used to describe each object) over the quality of detected clusters, real-world applications have been performed in the fields of territory characterization, mining of relations among social indicators and geomorphological features and air pollution analysis. In particular, referring to the territory characterization application, different runs

have been performed to estimate if and how the number of obtained clusters are affected by some learning parameters (such as the homogeneity threshold value) or by the dimension of data and data dimensionality. Results have shown that the number of clusters grows roughly linearly as either the threshold value or the dimension of data increases. Conversely, data dimensionality does not affect neither the number nor the shape of detected clusters.

In conclusion, the performed experiments have demonstrated that CORSO is worth to be applied to real-world domains, where it can be profitably used not only to detect groups of data, but also to provide intensional descriptions of them that can be used to explain the intrinsic sense of each cluster.

6.2 Future work

Different aspects concerning the proposed method have been dealt over this dissertation and some of them have given rise to some proposal for future works.

First of all, CORSO can be improved by making it able to deal with numerical relations. In fact, an important limitation of the current implementation of CORSO is that it can only cluster objects linked by binary relations. However, some datasets (such as the spatial ones) contain data naturally related by numerical relations (distance, for instance). Applying current version of CORSO to these datasets requires a transformation of numerical relations into corresponding binary ones. For instance, the distance relation D between two objects a and b can be transformed in the binary relation $B(a,b) = true \iff D(a,b) \leq threshold$. However, such transformation is quite restrictive for two reasons: (i) it requires to fix a threshold that could heavily affect the clustering results and (ii) it involves a kind of information loss when moving from numerical information of relations to binary ones.

Another improvement can concern the seed selection criteria adopted by the method: currently, CORSO selects the seed object according to one of the following criteria: sequential selection (SEQ), ascending order of connectivity (ASC) or descending order of connectivity (DOC). Choosing the right criteria for each application is not easy, since each of them stems from a different assumption whose suitability can vary according to the considered domain and the goal of experimentation. A proposal could be to extend the method with a further strategy that puts together all the above mentioned strategies performing a clustering task for each of them and that choose as final clustering result that one that is best scored according to some heuristic that evaluates the clusters quality. For instance, a possible heuristic for the evaluation of clusters quality could be based on the (weighted) mean of the homogeneity values associated with the calculated clusters.

Concerning the cluster construction, the detection of seed neighborhood as initial set of objects candidate to become cluster can sometimes become restrictive. In some case it happens that a natural cluster is not detected by CORSO simply because not all the neighbors of the chosen seed belong to the cluster. Currently, while in cluster expansion steps a single-neighbor check is performed when the neighbor-

hood homogeneity evaluation fails, only a neighborhood homogeneity evaluation is performed for the selected seed. On the other hand, it is clear that the single-neighbor check cannot been performed when evaluating the homogeneity of the seed neighborhood since we do not yet have a cluster to expand in this stage. Solution in this case can consist in considering as base step for clustering detection the *sub-neighborhoods* of the seed neighborhood (for instance, by removing one object at a time) when this one results to be not homogeneous.

The ARPA dataset discussed in Section 5.5 has suggested some other observations. Firstly, when all the neighbors of an objects o have already been clustered, we have observed that o generates a new cluster (composed only by o itself) instead of being labelled as noise. This happens because, in this case, the neighborhood of o is composed by only one object and, hence, is straighforwardly homogeneous (with homogeneity value equal to 1.0). This behaviour will be correct in the future versions of CORSO by introducing a check that will prevent to generate a new cluster when its cardinality is one. Secondly, in Section 5.5 we have suggested to improve the quality of induced models for datasets with a great quantity of numerical literals with an additional criterion for the selection of clauses that compose cluster models. Thirdly, we guess that a bottom-up hypothesis space exploration would lead to best clustering results than the top-down exploration currently adopted by ATRE since the induction of more complex theories would be more representative of information embedded into a cluster of objects. However, we do not linger over the last two observations since they are improvements concerning the ATRE learning engine rather than the clustering method discussed in this dissertation.

Finally, the entire clustering task could be enriched by a further a-posteriori step with the aim of performing a sort of cluster merging whenever two or more clusters are similar each other according their corresponding models.

Bibliography

[ABC+07] Oronzo Altamura, Margherita Berardi, Michelangelo Ceci, Donato Malerba, and Antonio Varlaro. Using colour information to understand censorship cards of film archives. *IJDAR*, 9(2-4):281–297, April 2007.

[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD Conference*, pages 49–60, 1999.

[Adh96] Junas Adhikary. Knowledge discovery in spatial databases: Progress and challenges. 1996.

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. pages 94–105, 1998.

[And01] Karl-Heinrich Anders. Data mining for automated gis data collection. *Wichmann Verlag, Heidelberg*, pages 263–272, 2001.

[And03] Karl-Heinrich Anders. A hierarchical graph-clustering approach to find groups of objects. 2003.

[Arb89] Giuseppe Arbia. *Spatial Data Configuration in Statistical Analysis of Regional Economics and Related Problems*. Kluwer Academic Publisher, Dordrecht, Holland, 1989.

[Ber02] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[BF00] Vladimir Batagelj and Anuška Ferligoj. Clustering relational data. In Wolfgang Gaul, Otto Opitz, and Martin Schader, editors, *Data Analysis*, pages 3–15. Springer-Verlag, 2000.

[BFR98] P. Bradley, U. Fayyad, and C. Reina. Scaling EM (Expectation Maximization) clustering to large databases, 1998.

[Bis92] Gilles Bisson. Conceptual clustering in a first order logic representation. In *ECAI*, pages 458–462, 1992.

[BRJD99] Hendrik Blockeel, Luc De Raedt, Nico Jacobs, and Bart Demoen. Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1):59–93, 1999.

[Bun88] Wray L. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artif. Intell.*, 36(2):149–176, 1988.

[BVM04] Margherita Berardi, Antonio Varlaro, and Donato Malerba. On the effect of caching in recursive theory learning. In *Inductive Logic Programming, 14th International Conference, ILP 2004*, pages 44–62, 2004.

[CA06] Michelangelo Ceci and Annalisa Appice. Spatial associative classification: propositional vs structural approach. *J. Intell. Inf. Syst.*, 27(3):191–213, 2006.

[CBP02] Rob Cross, Stephen P. Borgatti, and Andrew Parker. Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration. *California Management Review*, 44(2), 2002.

[CGT89] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.

[Cre91] Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, New York, 1991.

[DBK+99] S. Džeroski, H. Blockeel, S. Kramer, B. Kompare, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. In S. Džeroski and P. Flach, editors, *International Workshop on Inductive Logic Programming ILP 1999*, volume 1634 of *LNAI*, pages 80–91. Springer-Verlag, 1999.

[DD97] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine learning journal*, (2/3):99–146, 1997.

[DL01] S. Džeroski and N. Lavrač. *Relational Data Mining*, chapter An Introduction to Inductive Logic Programming, pages 48–73. LNAI. Springer-Verlag, 2001.

[EFKS00] Martin Ester, Alexander Frommelt, Hans-Peter Kriegel, and Jörg Sander. Spatial data mining: Database primitives, algorithms and efficient dbms support. *Data Mining and Knowledge Discovery*, 4(2/3):193–216, 2000.

[Ege91] M. Egenhofer. Reasoning about binary topological relations. In O. Gunther and H.J. Schek, editors, *Second Symposium on Large Spatial Databases*, volume 525, pages 143–160. Springer-Verlag, 1991.

[EKS01] Martin Ester, Hans-Peter Kriegel, and Jörg Sander. Algorithms and applications for spatial data mining. *Geographic Data Mining and Knowledge Discovery*, 5(6), 2001.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[EKSX98] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. Clustering for mining in large spatial databases. *KI-Journal*, 1, 1998.

[EL01] Vladimir Estivill-Castro and Ickjai Lee. Fast spatial clustering with different metrics and in the presence of obstacles. In *International Symposium on Advances in geographic information systems*, pages 142–147. ACM Press, 2001.

[EMS91] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. Flexible matching for noisy structural descriptions. In *International Joint Conference on Artificial Intelligence*, pages 658–664, 1991.

[EW96] Werner Emde and Dietrich Wettschereck. Relational instance based learning. In Lorenza Saitta, editor, *Machine Learning - Proceedings 13th International Conference on Machine Learning*, pages 122 – 130. Morgan Kaufmann Publishers, 1996.

[Fay96] Usama M. Fayyad. *Advances in Knowledge Discovery in Databases*. AAAI Press/MIT Press, Menlo Park, CA, 1996.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[Fla98] Peter A. Flach. The logic of learning: a brief introduction to inductive logic programming. In *Proceedings of the CompulogNet Area Meeting on Computational Logic and Machine Learning*, pages 1–17. University of Manchester, 1998.

[FSS96] Usama M. Fayyad, Gregory Piatetsky Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: an overview. *Advances in knowledge discovery and data mining*, pages 1–34, 1996.

[Fur99] Johannes Furnkranz. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13(1):3–54, 1999.

[GHC01] Jesus A. Gonzalez, Lawrence B. Holder, and Diane J. Cook. Graph-based concept learning. In *FLAIRS Conference*, pages 377–381, 2001.

[Got87] Georg Gottlob. Subsumption and implication. *Inf. Process. Lett.*, 24(2):109–111, 1987.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. pages 73–84, 1998.

[GRS00] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[GZ05] Peter A. Gloor and Yan Zhao. TeCFlow - A Temporal Communication Flow Visualizer for Social Network Analysis. 2005.

[HC03] Lawrence B. Holder and Diane J. Cook. Graph-based relational learning: current and future directions. *SIGKDD Explorations*, 5(1):90–93, 2003.

[HK98] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[HK01] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*, chapter Introduction, pages 1–38. Morgan Kaufmann Publishers, 2001.

[HKT01] Jiawei Han, Micheline Kamber, and Anthony K.H. Tung. Spatial clustering methods in data mining: A survey. pages 188–217, 2001.

[HN03] Zhexue Huang and Michael K. Ng. A note on k-modes clustering. *J. Classif.*, 20(2):257–261, 2003.

[HV03] E. Hancock and M. Vento. *Graph Based Representations in Pattern Recognitions*. Springer-Verlag, 2003.

[HWB01] Tamás Horváth, Stefan Wrobel, and Uta Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1-2):53–80, 2001.

[Jai06] Myank Jain. iQUEST Communication Processing Guide - Version 1.0.71, 2006.

[JCH01] Istvan Jonyer, Diane J. Cook, and Lawrence B. Holder. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2:19–43, 2001.

[KF05] Richard Kirkby and Eibe Frank. WEKA Explorer User Guide for Version 3-5-0, 2005.

[KHK99] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: A hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[KLF01] Stefan Kramer, Nada Lavrač, and Peter Flach. *Relational Data Mining*, chapter Propositionalization Approaches to Relational Data Mining, pages 262–291. LNAI. Springer-Verlag, 2001.

[Kol01] Erica Kolatch. Clustering algorithms for spatial databases: A survey. 2001.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[Kre01] Valdis E. Krebs. Connections. *Journal of the International Network for Social Network Analysis*, 24(3), 2001.

[KW98] Mathias Kirsten and Stefan Wrobel. Relational distance-based clustering. In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 261–270, London, UK, 1998. Springer-Verlag.

[LANP03] Onofrio Lattarulo, Lorenzo Angiuli, Alessandra Nocioni, and Vito M. Perrino. *Aria di citta'*. A.R.P.A. Puglia, 2003.

[LD94] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Maths and Statistics Problems*, volume 1, pages 281–297, 1967.

[Mal03] D. Malerba. Learning recursive theories in the normal ILP setting. *Fundamenta Informaticae*, 57(1):39–77, 2003.

[MAVL05] Donato Malerba, Annalisa Appice, Antonio Varlaro, and Antonietta Lanza. Spatial clustering of structured objects. In Stefan Kramer and Bernhard Pfahringer, editors, *Inductive Logic Programming, 15th International Conference, ILP 2005*, volume 3625 of *Lecture Notes in Computer Science*, pages 227–245. Springer, 2005.

[MD94] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[MEL+03] D. Malerba, F. Esposito, A. Lanza, F. A. Lisi, and A. Appice. Empowering a gis with inductive learning capabilities: The case of ingens. *Journal of Computers, Environment and Urban Systems, Elsevier Science*, 27:265–281, 2003.

[MF03] D. Mavroeidis and P.A. Flach. Improved distances for structured data. In T. Horváth and A. Yamamoto, editors, *Inductive Logic Programming, 13th International Conference*, volume 2835, pages 251–268. Springer-Verlag, 2003.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill Companies, Inc., 1997.

[MLBM03] Dunja Mladenic, Nada Lavrac, Marko Bohanec, and Steve Moyle. *Data Mining and Decision Support: Integration and Collaboration*, chapter Data Mining, pages 3–14. Kluwer Academic Publishers, 2003.

[Mug92] S. Muggleton. Inverting implication. In S. Muggleton, editor, *Proceedings of the 2nd International Workshop on Inductive Logic Programming*, pages 19–39, 1992.

[NAJ03] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[ND96] S-H. Nienhuys-Cheng and R. De Wolf. The subsumption theorem in inductive logic programming: Facts and fallacies. In Luc De Raedt, editor, *Advances in Inductive Logic Programming*, pages 265–276. 1996.

[NH94] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[Ora05] Oracle Spatial: User's Guide and Reference, 2005.

[OT79] S. Openshaw and P. J. Taylor. *Statistical applications in the spatial sciences*, chapter A million or so correlation coefficients: three experiments on the modifiable areal unit problem, pages 127–144. Wrigley N. Publishers, London, Pion, 1979.

[Pat91] D.W. Patterson. *Introduction to Artificial Intelligence and expert systems.* Prentice-Hall, 1991.

[Plo69] G.D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1969.

[Plo71] G. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press, 1971.

[Rig98] Fabrizio Riguzzi. *Extensions of Logic Programming as a Representation Language for Machine Learning.* PhD thesis, DEIS, Università of Bologna, November 1998. Technical Report DEIS-LIA-98-005, LIA Series n.33.

[RL96] Luc De Raedt and Nada Lavrač. Multiple predicate learning in two inductive logic programming settings. *Journal on Pure and Applied Logic*, 4(2):227–254, 1996.

[Sam95] Hanan Samet. Spatial data structures. In *Modern Database Systems: The Object Model, Interoperability, and Beyond.*, pages 361–385. 1995.

[SCZ98] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 428–439, 24–27 1998.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[SM86] Robert E. Stepp and Ryszard S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. *Machine Learning: An Artificial Intelligence Approach*, 2:471–498, 1986.

[SSV+02] S. Shekhar, P. R. Schrater, R.R. Vatsavai, W. Wu, and S. Chawla. Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on on Multimedia*, 4(2):174–188, 2002.

[Tob79] W. Tobler. Cellular geography. In S. Gale and G.Olsson, editors, *Philosophy in Geography*, 1979.

[Tou91] G. Toussaint. Some unsolved problems on proximity graphs. In D. Dearholt and F. Harary, editors, *First Workshop on Proximity Graphs*, 1991.

[VBM04a] Antonio Varlaro, Margherita Berardi, and Donato Malerba. Improving efficiency of recursive theory learning. In E. Panegai and G. Rossi, editors, *Italian Conference on Computational Logic, CILC*, pages 220–234, 2004.

[VBM04b] Antonio Varlaro, Margherita Berardi, and Donato Malerba. Learning recursive theories with the separate-and-parallel conquer strategy. In J. Fuernkranz, editor, *ECML/PKDD Workshop on Advances in Inductive Rule Learning*, pages 179–193, 2004.

[VD01] Wim Van Laer and Luc De Raedt. *Relational Data Mining*, chapter How to Upgrade Propositional Learners to First Order Logic: A Case Study, pages 235–261. LNAI. Springer-Verlag, 2001.

[Vis83] M. Visvalingam. Operational definitions of area based social indicators. *Environment and Planning*, A(15):831–839, 1983.

[Wek05] Weka 3.5.0, Java Programs for Machine Learning, 2005.

[WF00] Ian H. Witten and Eibe Frank. WEKA - Machine learning algorithms in Java, 2000.

[WH03] Xin Wang and Howard J. Hamilton. DBRS: A density-based spatial clustering method with random sampling. In *PAKDD*, pages 563–575, 2003.

[Wro01] S. Wrobel. *Relational Data Mining*, chapter Inductive logic programming for knowledge discovery in databases, pages 74–101. LNAI. Springer-Verlag, 2001.

[WYM97] Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *Twenty-Third International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece, 1997. Morgan Kaufmann.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. pages 103–114, 1996.