Philosophy Doctor Thesis

Jointly submitted at
The Institut National des Sciences Appliquées de Lyon
(Ecole Doctorale Informatique, Information pour la Société)
and the Universidad de las Américas, Puebla

by

Manuel Alfredo PECH PALACIO

December 12th, 2005

"Spatial Data Modeling and Mining using a Graph-based Representation"

PhD Committee Chair, Pr. XX, University of XX

Tutors:

Dr. David Ricardo Sol Martínez, UDLAP
Dr. Jesús A. González Bernal, INAOE
Pr. Robert Laurini, INSA of Lyon
Dr. Anne Tchounikine, INSA of Lyon

Reviewers:

Pr. YY, University of YY
Dr. ZZ, University of ZZ

Examiner:

Dr. YY, University of YY
Dr. ZZ, University of ZZ

ABSTRACT

Motivation

Several approaches have been developed for mining spatial data (i.e. generalization-based, clustering, spatial associations, approximation and aggregation, mining in image and raster databases, spatial classification and spatial trend detection). However, we argue that these approaches do not consider all the elements found in a spatial database (spatial data, non-spatial data and spatial relations among the spatial objects) in an extended way. Some of them focus first on spatial data and then on the non-spatial data or vice versa, and others consider restricted combinations of these elements. We think that it is possible to enhance the generated results of the data mining task by mining them as a whole and not as separated elements (they are related elements). A graph representation provides the flexibility to describe these elements together and this is the motivation to explore the area of graph-based spatial knowledge discovery.

Proposal

Our idea is to create a unique graph-based model to represent spatial data, non-spatial data and the spatial relations among spatial objects. We will generate datasets composed of graphs with a set of these three elements. We consider that by mining a dataset with these characteristics a graph-based mining tool can search patterns involving all these elements at the same time improving the results of the spatial analysis task. A significant characteristic of spatial data is that the attributes of the neighbors of an object may have an influence on the object itself. So, we propose to include in the model three relationship types (topological, orientation, and distance relations).

In the model the spatial data (i.e. spatial objects), non-spatial data (i.e. non-spatial attributes), and spatial relations are represented as a collection of one or more directed graphs. A directed graph contains a collection of vertices and edges representing all these elements. Vertices represent either spatial objects, spatial relations between two spatial objects (binary relation), or non-spatial attributes describing the spatial objects. Edges represent a link between two vertices of any type. According to the type of vertices that an edge joins, it can represent either an attribute name or a spatial relation name. The attribute name can refer to a spatial object or a non-spatial entity. We use directed edges to represent directional information of relations among elements (i.e. object *x* touches object *y*) and to describe attributes about objects (i.e. object *x* has attribute *z*).

We propose to adopt the Subdue system, a general graph-based data mining system developed at the University of Texas at Arlington, as our mining tool. Subdue discovers

substructures using a graph-based representation of structural databases. The substructures (a connected subgraph within the graphical representation) describe structural concepts in the data (i.e. patterns). The discovery algorithm is a computationally constrained beam search. The algorithm begins with the substructure matching a single vertex in the graph. Each iteration, the algorithm selects the best substructure and incrementally expands the instances of the substructure. An instance of a substructure in the input graph is a subgraph that matches (graph theoretically) that substructure.

A special feature named overlap has a primary role in the substructures discovery process and consequently a direct impact over the generated results. However, it is currently implemented in an orthodox way: all or nothing. If we set overlap to true, Subdue will allow the overlap among all instances sharing at least one vertex. On the other hand, if overlap is set to false, Subdue will not allow the overlap among instances sharing at least one vertex. So, we propose a third approach: limited overlap, which gives the user the capability to set over which vertices the overlap will be allowed (vertices representing remarkable elements that refer, for instance, to a spatial object in a spatial database or to some characteristic defining a particular topic of a dataset). We visualize directly three motivations issues to propose the implementation of the new algorithm: search space reduction, processing time reduction, and pattern oriented search.

Contribution

The contribution to the discovery knowledge in the spatial data domain, described in this dissertation, is the development of a new approach for spatial data modeling and mining using a graph-based representation. This contribution includes the following results:

- New graph-based data representation for spatial, non-spatial data and spatial relations.
- New algorithm to discover substructures using a limited overlap approach in the Subdue system.
- A prototype system implemented the proposed model.

Table of contents

List of figures

List of tables

Chapter 1

INTRODUCTION

Due to the advances in data generation and recompilation we are facing a continuous growth in the data collections. The analysis and interpretation of this data by manual techniques is sometimes a tough task; therefore, different methods have been proposed to help us transform it into useful knowledge. Knowledge discovery in databases (*KDD*) is defined as the non-trivial extraction of implicit, previously unknown, and potentially useful information from data [16]. This is an interactive and iterative process that involves several phases. The data mining phase is the nucleus of the process; it consists of the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produces a particular enumeration of patterns over the data [12]. Data mining involves the integration of methods from different scientific fields like machine learning, database technology and statistics. The first approaches developed focused in the discovery of knowledge from relational data.

Nowadays, however, terms like geoprocessing and Geographic Information System are used widely in daily life. This is a result of the improvement in the human capabilities to create, manipulate, store and use data from phenomena on, above or below the earth's

surface. This data is known as spatial data. Spatial data mining focuses in the discovery of implicit and previously unknown knowledge in spatial databases [16]. Spatial data has many features that distinguish it from relational data such as the relationships among the objects, complexity, and the query language used to access them.

1.1 Motivation

As result of the growing in the volume of spatial datasets and the necessity for tools to help us transform them into useful information, several approaches have been developed for knowledge discovery from spatial data: the basic in the *Generalization-based* methods [20][34] is that data and objects often contain detailed information at primitive concept levels but sometimes it is desirable to summarize that information and present it at a higher concept level. *Clustering* [25][27][36][38][44] is the process of grouping physical or abstract objects into classes (clusters) of similar objects so that the members of a cluster are as similar as possible whereas the members of different clusters differ as much as possible from each other. *Spatial associations* [13][28] discover rules that associate one or more spatial objects with other spatial objects. In *Approximation and aggregation* methods [26] the idea is to analyze the characteristics of the clusters in terms of the features (objects) close to them. Aggregate proximity is the measure of closeness of the set of points in the cluster to a feature. *Mining in image and raster databases* [13][14] can be viewed as another approach of spatial data mining. Some applications of this approach (based on images) are automatic recognition and categorization of astronomical objects; classification of stars, galaxies and other stellar objects. *Spatial Classification* [30] is the task of

assigning objects to a set of classes based on their attribute values. *Spatial Trend Detection* [10] describes the regular change of one or more non-spatial attributes of an object when moving away from a given starting object.

However, we argue that these approaches do not consider all the elements found in a spatial database (spatial data, non-spatial data and spatial relations among the spatial objects) in an extended way. Some of them focus first on spatial data and then on the non-spatial data or vice versa, and others consider restricted combinations of these elements. We think that it is possible to enhance the generated results of the data mining task by mining them as a whole and not as separated elements (in the real world they are related). In this context, we propose to use a graph-based representation since it provides the flexibility to describe these elements together and this is the motivation to explore the area of graph-based spatial knowledge discovery.

Our work is based in the hypothesis if we create a graph-based model to represent together spatial and non-spatial data and if we use this model for generating a dataset composed of both type of data, then we can apply data mining techniques using this knowledge representation to spatial and non-spatial data at the same time and get enriched results (patterns found through data mining) considering both kind of data about objects and the spatial relations among them.

1.2 Proposal

Our proposal is to create a unique graph-based model to represent spatial data, non-spatial data and the spatial relations among spatial objects. We will generate datasets composed of graphs with a set of these three elements. We consider that by mining a dataset with these characteristics a graph-based mining tool can search patterns involving all these elements at the same time improving the results of the spatial analysis task. A significant characteristic of spatial data is that the attributes of the neighbors of an object may have an influence on the object itself. We propose to include in the model three relationship types (topological, orientation, and distance relations).

In the model the spatial data (i.e. spatial objects), non-spatial data (i.e. non-spatial attributes), and spatial relations are represented as a collection of one or more directed graphs. A directed graph contains a collection of vertices and edges representing all these elements. Vertices represent either spatial objects, spatial relations between two spatial objects (binary relation), or non-spatial attributes describing the spatial objects. Edges represent a link between two vertices of any type. According to the type of vertices that an edge joins, it can represent either an attribute name or a spatial relation name. The attribute name can refer to a spatial object or a non-spatial entity. We use directed edges to represent directional information of relations among elements (i.e. object *x* touches object *y*) and to describe attributes about objects (i.e. object *x* has attribute *z*).

We propose to adopt the Subdue system [23][43], a general graph-based data mining system developed at the University of Texas at Arlington, as our mining tool. Subdue

discovers substructures using a graph-based representation of structural databases. The substructures (a connected subgraph within the graphical representation) describe structural concepts in the data (i.e. patterns). The discovery algorithm is a computationally constrained beam search. The algorithm begins with the substructure matching a single vertex in the graph. Each iteration, the algorithm selects the best substructure and incrementally expands the instances of the substructure. An instance of a substructure in the input graph is a subgraph that matches (graph theoretically) that substructure.

A special feature named overlap has a primary role in the substructures discovery process and consequently a direct impact over the generated results. However, it is currently implemented in an orthodox way: all or nothing. If we set overlap to true, Subdue will allow the overlap among all instances sharing at least one vertex. On the other hand, if overlap is set to false, Subdue will not allow the overlap among instances sharing at least one vertex. So, we argue a third option is needed: a limited overlap. With this option we give the user the capability to set over which vertices the overlap will be allowed (vertices representing remarkable elements that refer, for instance, to a spatial object in a spatial database or to some characteristic defining a particular topic of a dataset). We visualize directly three motivations issues to propose the implementation of the new algorithm: search space reduction, processing time reduction, and pattern oriented search.

1.3 Contribution

The contribution to the discovery knowledge in the spatial data domain, described in this thesis, is the development of a new approach for spatial data modeling and mining using a graph-based representation. This approach includes the following issues:

- We proposed a new graph-based data representation for spatial data, non-spatial data and spatial relations among the spatial objects. We visualize two objectives for creating a data model with these characteristics. The first one is to create a unique graph-based dataset representing these related elements. The second one is to use this dataset to feed a graph-based mining system, so we can discover single patterns (involving these elements) which help us to describe/understand the data, based on the premise, they are related elements in the real world.
- We proposed a new algorithm to discover substructures (patterns) using a limited overlap approach in the Subdue system. We visualize directly three motivations issues to propose the implementation of the new algorithm: search space reduction, processing time reduction, and specialized overlapping pattern oriented search.
- We designed and developed a prototype system implementing the proposed model. The prototype provides to the user a friendly graphical user interface for managing the spatial layers to work with, for creating spatial and non-spatial graphs, for mining those graphs and for displaying the generated results.

1.4 Organization of the thesis

The thesis is structured in the following way: Chapter 1 presents the motivation, proposal, and contributions of the thesis. Related work is described in chapter 2. In chapter 3 we detail our graph-based model to represent together spatial data, non-spatial data, and spatial relations. Our graph-based data mining tool, the Subdue system, and the new limited overlap algorithm are described in chapter 4. A prototype system implementing our model is presented in chapter 5. Use-cases showing the applicability of our proposal are described in chapter 6. Finally, conclusions and final remarks are commented in chapter 7.

Chapter 2

RELATED WORK

Knowledge Discovery in Databases, spatial data, non-spatial data, Data Mining, Spatial Data Mining, Geographic Information System, geoprocessing, and Geomatics are terms broadly used nowadays. This chapter presents an outline related to these issues. The Geographic Information Systems are presented in section 2.1. In section 2.2 we describe related work about Knowledge Discovery in Databases and data mining. Finally, section 2.3 introduces the three types of spatial relations we propose to incorporate in our model to represent spatial data.

2.1 Geographic Information System (GIS)

A Geographic Information System is defined as a tool for the manipulation of geographic data [2]. The *GIS* performs a great diversity of functions; some of them are the compilation, verification, storage, retrieval, manipulation, update and visualization of geographic data. Additionally, one of its more important features is the inclusion of data analysis modules.

All these functions are applied by a *GIS* to geographic data, stored generally in a geographic database. The data processed and manipulated is *georeferenced*, that is, it is assigned to a specific location on the Earth's surface using a coordinate system.

The *GIS* can process data from several sources. For example, data collected from maps, images and photography, statistical data from mathematical analyses, and data from *CAD* systems (Computer-Assisted Design).

A *GIS* organizes and handles digital data stored generally in a geographic database. The databases are important in the *GIS* technology because they store geographic data with a structured form, allowing the data to be used for many tasks. Many *GIS* implement additional functionalities when they use database management systems (*DBMS*) to store and to handle all or some data in an independent subsystem.

The diversity of the uses of the *GIS* has generated the proliferation of a great variety of definitions of *GIS*. A user generally defines a *GIS* according to what he/she uses it for and his/her own experience and abilities. Some of these definitions are:

- A system for data processing designed for the production and/or visualization of maps.
- An information system to respond to questions about Earth's properties or soil types.
- A system for decision making support in situations of natural phenomena.
- An electronic positioning system to be used by terrestrial or marine transportation.

The *GIS* frequently is named according to its application field [2]. For example, when they are used to manage land's registries they are generally called Land Information Systems (*LIS*); in applications of municipal and natural resources they are important components of the Urban Information Systems (*UIS*), and Natural Resources Information Systems (*NRIS*) respectively. The term Automatic Mapping/Facility Management (*AM*/*FM*) is used by public maintenance companies, transportation agencies, and local governments for systems dedicated to the operation and maintenance of networks.

The *GIS*'s field (see Figure 2.1) involves many disciplines, applications, data types, and end users, for example:

- Disciplines: Computer Science, Cartography, Spatial Analysis, Topography, Hydrography, Statistic, Sciences of the Information, Planning, etc.
- Applications: Operation and maintenance of networks and, other devices, administration of natural resources, highway planning, map production, urban analysis, planning analysis, etc.
- Data: Digital maps, digital images and photography, data satellites, video images, etc.
- Users: Planners, topographers, vulcanologists, geographers, environmentalist, engineers, etc.

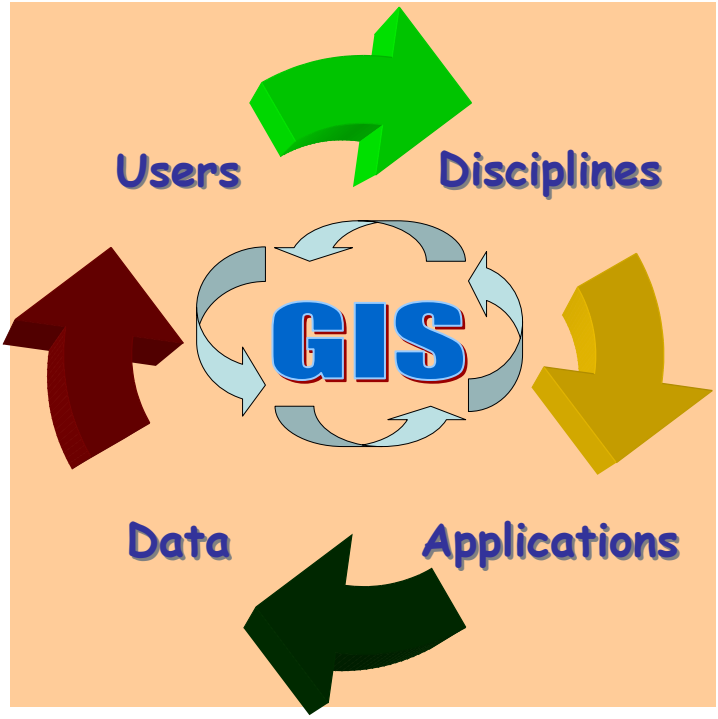


Figure 2.1. Geographic Information System.

Geographic Information Systems involve the uses of systems and science. Their use arise question such as: How does a *GIS* user know that the results obtained are accurate? How can user interfaces be made readily understandable by novice users? Goodchild M. F. published in 1992 a paper where he argued that questions such as these and their systematic study constituted a science.

Geoprocessing study the fundamental issues arising from geographic information (i.e. creation, handling, storage and use of the information). The term *Geomatics*, the fusion of ideas from geosciences and informatics, is defined as the umbrella covering all fields that are today important for understanding and further developing information systems in this context [31][32][33].

A *GIS* offers its users greater capabilities to process datasets than those offered by manual systems. In a *GIS* database the data are stored in a structured form, unlike the manual systems where the data are stored in files, maps, and/or reports. The data can be recovered from geographic databases and processed faster and more safely than in the manual systems.

We can classify the *GIS*'s users into two groups. In the first group, we find professional operators who spend a lot of time working with *GIS* technology. They are people trained in some particular software and they know the capabilities of this technology. Many times these people do not use the results of their work, but they pass them on to the end users.

The second user group spends less time working with the *GIS*'s. They maintain geographic information in order to have tools which help them in the decision making process. There are few opportunities for extensive training in the GIS tools, and consequently the *GIS* must be simple and easy to handle.

2.2 Data Mining

Knowledge Discovery in Databases (*KDD*) is be defined as the non-trivial extraction of implicit, previously unknown, and potentially useful information from data [16]. This is an interactive and iterative process that involves several phases: data preparation, search for

patterns over the data, evaluation and interpretation of discovered patterns, and refinement of the whole process as we shown in Figure 2.2.

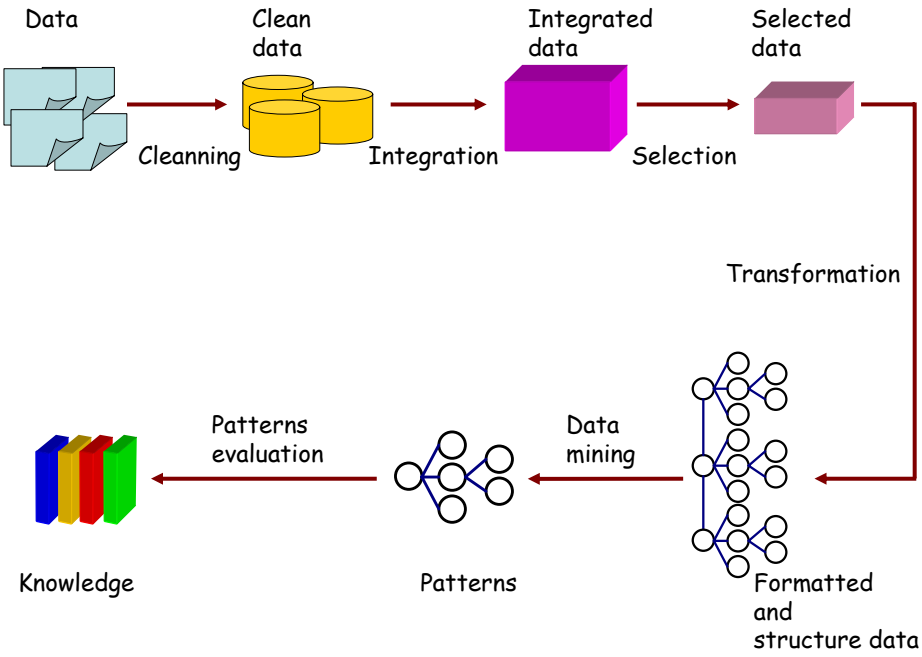


Figure 2.2. Knowledge Discovery in Databases.

The data mining phase (the search for patterns) is the nucleus of the process; it consists of the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produces a particular enumeration of patterns over the data [4][12].

Data mining involves the integration of methods from different scientific fields such as machine learning, database technology, statistics, and visualization as we shown in Figure 2.3. The first approaches developed focused in the discovery of knowledge from relational data.

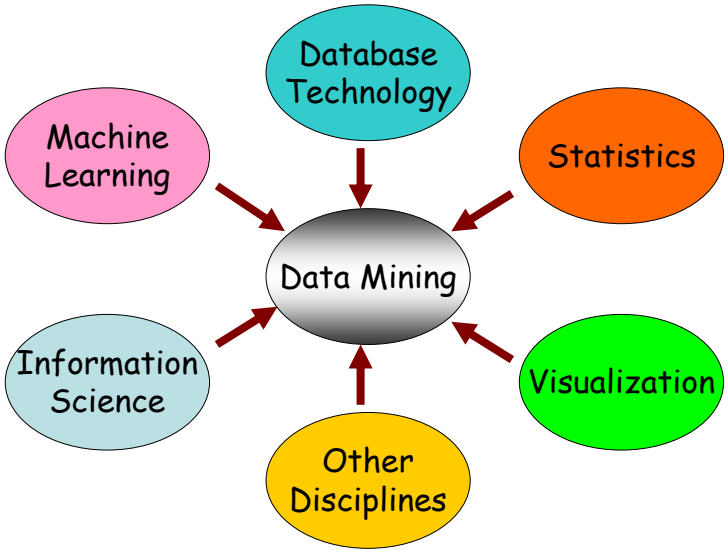


Figure 2.3. Data mining: integration of several fields.

Several architectures have been proposed for data mining [21][24]. Figure 2.4 presents an architecture based on the proposal of Han et al. [22]. The *user* is the trigger of the entire process and receptor of the discovered knowledge. *Data* may be fetched from several sources such as files, databases, and data warehouses using a *data server* module. The *data mining engine* may use one or more data mining techniques for searching patterns from data. The significance, importance and interestingness of the found patterns are evaluated by the *pattern evaluation* module. The *data mining engine* and *pattern evaluation* modules may use background knowledge stored in a *knowledge database*. The role of the *graphical user interface* is to receive the user requirements and to deliver the generated results. Since this is an iterative and interactive process, the components may interact among themselves.

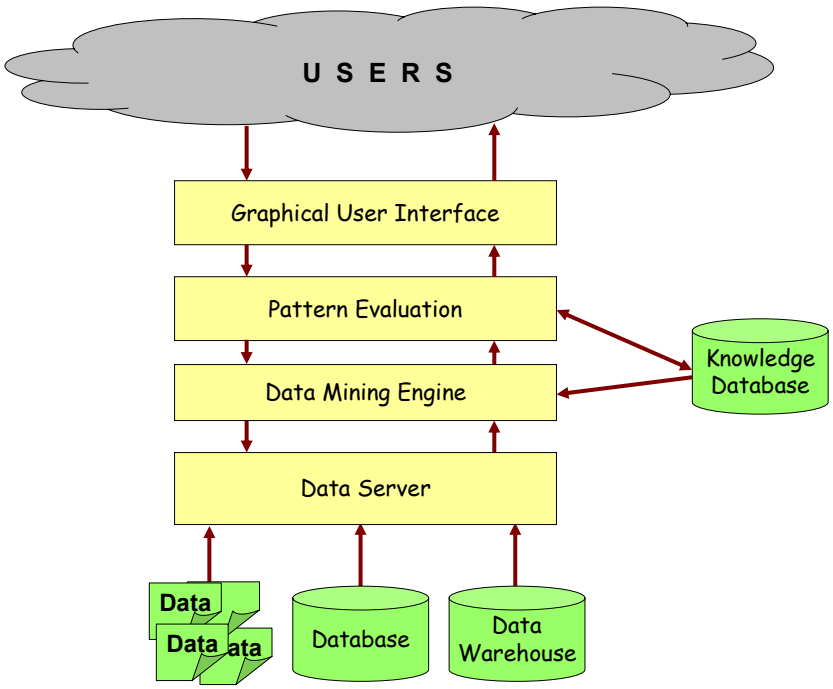


Figure 2.4. Architecture for a KDD system.

2.2.1 Spatial Data Mining

Climate change, natural risk prevention, human demography, deforestation, and natural resources atlas are examples from a large variety of issues arisen as result of the interaction among people and their natural environment, the earth planet. Data generated from those issues are known as spatial data. Spatial data mining methods focuses in the discovery of implicit and previously unknown knowledge in spatial databases [16].

Spatial data have many features that distinguish them from relational data. For example, the spatial objects may have topological, distance, and direction information, the complexity,

and the query language used to access them. Different approaches have been developed for knowledge discovery from spatial data such as Generalization-based method [20][34], Clustering [25][27][36][38][44], Spatial associations [13][28], Approximation and aggregation [26], Mining in image and raster databases [13][14], Classification learning [30], and Spatial trend detection [10]. In the following subsections we present a description of these spatial data mining approaches.

2.2.1.1 Generalization-based Method

Generalization has been shown to be one of the effective methods of discovering knowledge. It was introduced by the machine learning community and it is based on learning from examples techniques. The generalization based knowledge discovery requires concept hierarchies (given explicitly by experts or generated automatically). In the case of the spatial databases, there can be two types of concept hierarchies:

- Thematic hierarchies. We can generalize tomatoes and bananas as fruits, fruits and vegetables as cash crops.
- Spatial hierarchies. We can generalize some geographic points as a country or a region.

The approach introduced by the machine learning community (tuple-oriented) cannot be directly adopted for large spatial databases because it does not handle very well the noise and inconsistent data, and the algorithms are exponential in the number of examples. Han et al. [20] present a modified technique named attribute-oriented induction for mining

relational data. Lu et al. [34] extend attribute-oriented induction technique to spatial databases. Attributed-oriented induction is performed by climbing the generalization hierarchies and summarizing the general relationships between spatial and non-spatial data at a higher concept level. The authors present two generalization based algorithms:

- **Non-spatial data dominant.** This method performs attribute-oriented induction on the non-spatial attributes, first generalizing them to a higher concept level and later merges corresponding spatial attributes (using spatial merge and approximation).
- **Spatial data dominant.** Given the spatial data hierarchy, generalization can be performed first on the spatial data and then generalizing their corresponding non-spatial attributes.

Both algorithms assume that the rules to be mined are general data characteristics (characteristics rules) and that the discovery process is initiated by the user who provides a learning request. A disadvantage in this approach is the case where a hierarchy may not exist or the hierarchy given by the experts may no be entirely appropriate in some cases. The quality of mined characteristics is highly dependent on the structure of the hierarchy.

2.2.1.2 Clustering

Clustering is the process of grouping physical or abstract objects into classes of similar objects. Clustering analysis helps construct meaningful partitioning of large set of objects. Due to the huge amount, characteristics and nature of spatial data, there are important challenges for a clustering algorithm:

- Achieve good time efficiency.

- Identify clusters irrespective of their shape or relative positions.
- Handling of noise or outliers.
- Handling data with large number of features (high dimensionality).

Data clustering identifies clusters, or densely populated regions, according to some distance measurement in a multidimensional dataset. Given a set of multidimensional data points, the data space is usually not uniformly occupied by the data points. Data clustering identifies the sparse and the crowded places, and hence discovers the overall distribution patterns of the dataset. We can classify clustering algorithms in four main approaches: Partitioning, Hierarchical, Locality-based and Grid-based algorithms.

- Partitioning algorithms partition a database of n objects into a set of k clusters which are represented by the gravity of the cluster (k-means algorithms) or by one representative object of the cluster (k-medoid algorithms). These algorithms use a two-step procedure. First, they determine k representatives, next, assign each object to the cluster with its representative closest to the considered object.
- Hierarchical clustering algorithms decompose the database into several levels of partitioning which are usually represented by a *dendrogram*. The algorithm iteratively splits the database into smaller subsets until some termination condition is satisfied. The *dendrogram* can either be created *top-down* (divisive) or *bottom-up* (agglomerative).
- Locality-based clustering algorithms group neighboring data elements into clusters based on local conditions and therefore allow the clustering to be performed in one scan of the database.

- Grid-based algorithms quantize the space into a finite number of cells and then do all operations on the quantized space. They frequently use hierarchical agglomeration as one phase of processing.

Classification of clustering algorithms is neither straightforward, nor canonical. Some algorithms perform clustering by combining techniques from these approaches. Important issues in clustering algorithms include the following properties:

- The algorithm must be efficient (time complexity).
- Ability to handle noise (outliers).
- Sensibility to data input order.
- Priori knowledge and parameters.
- Ability to find clusters of arbitrary shape.
- Scalability to large databases.
- Ability to work with high dimensional data.

2.2.1.3 Spatial Associations

A spatial association rule is a rule which describes the implication of one or a set of features by another set of features in spatial databases [28]. An example of a spatial association rule is "the biggest industries in Mexico are close to DF".

A spatial association rule is of the form $X \rightarrow Y$, where X and Y are sets of spatial or non-spatial predicates. There are various kinds of spatial predicates that could constitute a

spatial association rule. Some examples are topological relations such as *intersects*, *overlap*, *disjoint*; and spatial orientations such as *left_of*, *west_of*.

Koperski and Han [28] developed an algorithm for spatial associations rules in spatial databases. They use the concepts of minimum support and minimum confidence introduced by Agrawal et al. [1] to develop associations rules from large database transaction database. The support of a pattern *A* in a set of spatial objects *S* is the probability that a member of *S* satisfies pattern *A*, and the confidence of $A \rightarrow B$ is the probability that a pattern *B* occurs if pattern *A* occurs. A user or an expert may satisfy thresholds to confine the rules to be discovered to be strong ones.

Although many spatial association rules may exist in large databases, some of them may occur rarely or may not hold in most cases. Also, such rules are usually not 100% true, but carry some non trivial knowledge.

The authors employ a method which uses a top-down progressive deepening search technique. The technique firstly searches at a high concept level for large patterns and strong implications relationships among the large patterns at a coarse resolution scale. Then only for those large patterns, it deepens the search to lower concept levels. Such a deepening search the process continues until no large patterns can be found. The search employed for large patterns at high concept levels is applied at a coarse resolution scale efficiently by using approximate spatial computation algorithms such as *R-trees* or *plane-sweep* techniques operating on minimum bounding rectangles (*MBR*). Only the candidate spatial predicates, which are detailed reviewed, will be computed by refined spatial

techniques. Such multiples-level approach saves much computation because it is very expensive to perform detailed spatial computation for all possible spatial association relationships.

2.2.1.4 Approximation and Aggregation

Clustering algorithms are effective and efficient methods for answering questions such as: where are the clusters in the spatial database? In some cases it is also important to answer the question why the clusters are there. We can rephrase the question as: what are the characteristics of the clusters in terms of the features (objects) that are close to it? Knorr and Ng [26] present a study based in this question.

The aggregate proximity is the measure of closeness of the set of points in the cluster to a feature as opposed to the distance between a cluster boundary and the boundary of a feature. Finding aggregate proximity relationships is not as simple as it may seem. There are three reasons:

- The sizes and shapes of the cluster and the features may vary greatly.
- There may be a very large number of features to examine.
- Even if a suitable feature (i.e. polygon) is found to describe the shape of the cluster of points, it is inappropriate to simply report those features whose boundaries are closest to the cluster's boundary, because the distribution of points in a cluster may not be uniform.

The authors propose the use of computational geometry concepts to find out the characteristics of a given cluster in terms of the features close to it. They present the algorithm *CRH* (where *C* is for encompassing circle, *R* for isothetic rectangle, and *H* for convex hull) which uses concepts as filters to reduce the candidate features at multiple levels. In general, they collect a large number of features from multiples sources (i.e. maps) and feed them along with the cluster to the algorithm *CRH* and discover knowledge about spatial relationships.

Approximation by circles and then by rectangles is used to eliminate features that have large aggregate distance to the cluster. After these filters, the algorithm calculates the aggregate proximity of points in the cluster to the convex boundary of each feature that passed through the previous filters. In the last step, the algorithm reports the features with the best aggregate proximities showing the minimum and maximum distances of points in the cluster to the feature, average distance, and percentages of points located in the distance less than specified threshold.

2.2.1.5 Mining an Image Database

Knowledge mining from image databases can be viewed as a special case of spatial data mining. For example, Fayyad et al. present a system [14] to identify and categorize volcanoes on the surface of Venus from images taken by the Magellan spacecraft. Three basic components are implemented in the system: data focusing, feature extraction, and classification learning. The first component increases the overall efficiency of the system by first identifying the portion of the image being analyzed that is most likely to contain a

volcano. They compare the intensity of the central pixel of a region to the estimated mean background intensity of its neighborhood pixels. The second component extracts interesting features from the data. The final component uses training examples provided by the experts to create a classifier that can discriminate between volcanoes and false alarms. For this task the authors implement decision trees.

Other studies of mining in image and raster databases are: Second Palomar Observatory Sky Survey [15], it uses decision trees for the classification of galaxies, stars and other stellar objects. Stolorz and Dean [41] proposed a system for detecting earthquakes from space. They combined methods of statistical interference, massively parallel computing, and global optimization to build the system that analyze tectonic activities with sub-pixel resolution over a large area. Stolorz et al. [42] and Shek et al. [39] carry out studies about fast spatio-temporal data mining from geophysical datasets; they described CONQUEST, a distributed parallel querying and analysis mining tool.

2.2.1.6 Classification Learning

Spatial classification has as objective to find rules that divide a set of objects into a number of groups, where objects in each group belong mostly to one class. Many types of information can be used to characterize spatial objects. We can classify such information into non-spatial attributes of objects, spatially related attributes with non-spatial values, spatial predicates and spatial functions.

Each of these categories may be used to extract both for class label attributes (attributes that divide data into classes) and predicting attributes (attributes on whose values the decision tree is branched). It is possible uses aggregate values for some of these attributes.

Koperski et al. [30] proposed and evaluated a method for classification of spatial objects. The method enables classification of spatial objects based on aggregate values of non-spatial attributes for neighboring regions. Spatial relations between objects on the map are taken into a count, which may be represented into the form of predicates.

2.2.1.7 Spatial Trend Detection

Spatial trend detection is defined as a regular change of one or more non-spatial attributes in the neighborhood of some object in a database [10]. An example of spatial trend is as "moving away from downtown Puebla, the price of land decreases".

Neighborhood paths starting from some point *x* are used to model the movement and a regression analysis is performed on the respective attribute values for the objects of a neighborhood path to describe regularity of change. In the regression analysis the distance from *x* is the independent variable and the difference of the attribute values are the dependent variable(s). There are two types of trends: global trends and local trends. In the first one, the existence of a global trend for a start object *x* indicates that if considering all objects on all paths starting from *x* the values for the specified attribute(s) in general tend to increase or decrease with increasing distance. Local trends exist only in a particular direction.

2.3 Spatial relations

The explicit location and extension of objects define implicit relations of spatial neighborhood. The information about the neighborhood of spatial objects constitutes a valuable element that must be considered in the mining task. In the following subsection we will present the Neighborhood Graphs, Neighborhood Paths and Neighborhood Indices concepts which introduce us to the three types of spatial relations we propose to include in our model to represent together spatial data.

2.3.1 Neighborhood Graphs, Neighborhood Paths and Neighborhood Indices

Martin Ester et al. [9][11] introduce the concept of neighborhood graphs for explicitly representing those implicit neighborhood relations. Neighborhood graphs may cover one of the following neighborhood relations:

- Topological.
- Distance.
- Direction.

These relations are called binary relations since we can determine spatial relations between pairs of objects.

A neighborhood graph G for some spatial relation neighbor is a graph where nodes are objects in the database and edges between nodes n_1 and n_2 represent the fact that the

relationship neighbor (n_1, n_2) holds. The neighborhood path in the neighborhood graph G is a set of nodes directly connected through the edges of the graph. The neighborhood index is a data structure that allows for the efficient execution of the operations for the construction of graph and for browsing and expansion of paths. It stores all neighbors for the objects in a database.

2.3.2 Topological Relations

Topological relations are those relations which are invariant under linear transformations, i.e. if both objects are rotated, translated or scaled simultaneously the relations are preserved. They present a definition of topological relations derived from the nine intersection model [6][7][8].

In the model (see Figure 2.5), the topological relations between two objects A and B are defined in terms of the intersections of object *A's* interior (A^o), object *A's* boundary (∂A) and object *A's* exterior (A^-) with object *B's* interior (B^o), object *B's* boundary (∂B) and object *B's* exterior (B^-). The exterior of an object is represented by its complement.

$$\begin{pmatrix} A^o \cap B^o & A^o \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Figure 2.5. Nine intersection model.

The topological relations between two objects are *disjoint*, *contains*, *inside*, *equal*, *touch***,** *covers*, *coveredBy*, and *overlap* as we show in Figure 2.6.

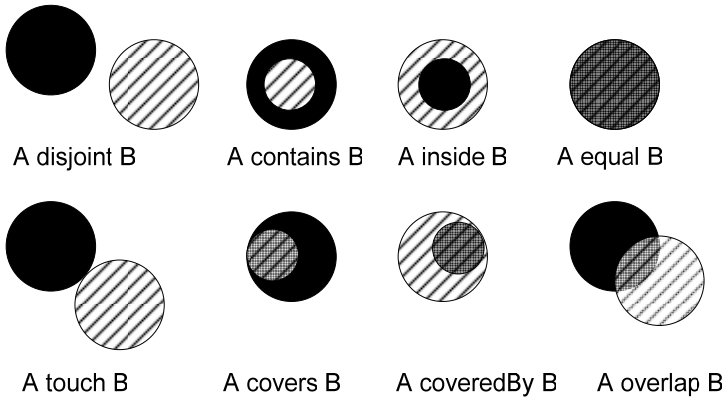


Figure 2.6. Topological Relations.

2.3.3 Distance Relations

Distance relations compare the distance between two objects with a given constant using arithmetic operators such as <, >, =. The distance between two objects is defined as the minimum distance between them (i.e. select all elements inside a radio of 50 km from a "*x*" point). Figure 2.7 shows two examples of this type of relation; in the figure we are representing how close and how far two objects are each other.

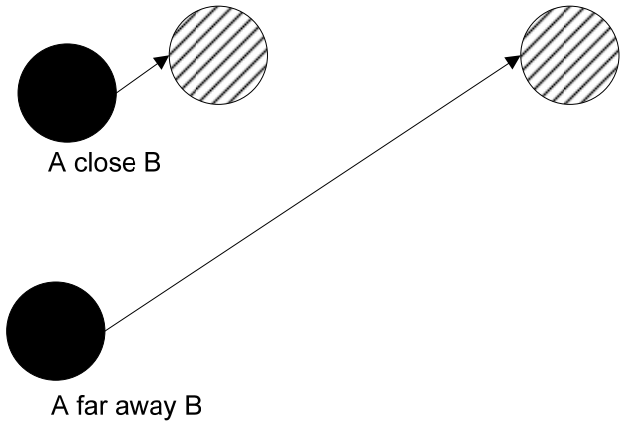


Figure 2.7. Distance Relations.

We propose to use Euclidian distance which is defined as the straight line distance between two points. In a plane with *p1* at (x_1, y_1) and *p2* at (x_2, y_2), the Euclidian distance is $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

2.3.4 Direction Relations

The direction relation *B R A* of two spatial objects *A* and *B* is defined using one representative point of object *A* and all points of the destination object *B*. It is feasible to define several possibilities of direction relations depending on the number of points that are considered in the source and destination objects. The representative point of a source object may be the center of the object or a point on its boundary. This representative point is used as the origin of a virtual coordinate system and its quadrant defines the direction.

The Direction relations between two objects are *North_of*, *South_of*, *East_of*, *West_of*, *Northeast_of*, *Northwest_of*, *Southeast_of*, and *Southwest_of*. In Figure 2.8 we show some examples of direction relations, for instance, object *D* is South of object *C* and East of object *A*.

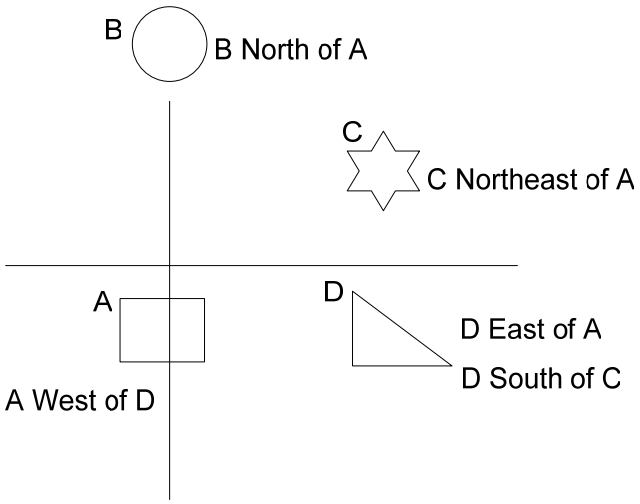


Figure 2.8. Direction Relations.

2.4 Conclusion

Data mining is a younger and promissory research field. Many of the data mining approaches developed for knowledge discovery in relational databases were extended to the spatial databases domain. In this chapter we presented several approaches such as generalization, clustering, spatial association, and spatial classifications. We have described three types of spatial relations that will be included in our model to represent as a unique graph-based dataset spatial data, non-spatial data and spatial relations.

In the next chapter we will describe the model. First, we will talk about the graph-based knowledge discovery, next we will detail our methodology, and finally we will present an example showing its applicability and generated results.

Chapter 3

GRAPH-BASED REPRESENTATIONS

This chapter describes our graph-based model to represent together spatial data, non-spatial data and spatial relations among the spatial objects. We propose to include three types of spatial relations: topological, distance and direction. Section 3.1 presents some issues related to the graph-based knowledge discovery. Our methodology is detailed in section 3.2. In section 3.3 we describe five graph-based representations based in our model. Finally, an example of the applicability of the proposal is presented en section 3.4.

3.1 Generalities

In chapter 2 we presented several approaches developed to search knowledge in spatial databases. The differences between these approaches are based on the data representation and the data mining algorithm used in the search task. Certainly, the data representation used by a mining tool is very important, and it has to be powerful enough to represent domains containing complex relations among their components (i.e. spatial data domain).

A graph-based representation has these characteristics [3][5][17][35]. It has the benefits of being easy to understand and flexible enough to create different representations of the same domain. The domain (i.e. data and relationships) is described using graphs. These graphs become the input to a graph-based discovery tool which uses a heuristic to choose the subgraphs that are considered important or discovered knowledge.

A graph is defined as a pair $G = (V,E)$. $V = \{v_0,\ldots,v_n\}$ denotes a finite set of elements called vertices. E is a set of edges satisfying $E \in f$: $V^2 \rightarrow V^2$ where f is an injective function. Then, each edge $e \in E$ is a pair (v_i,v_j). If (v_i,v_j) is an ordered pair for any $(v_i,v_j) \in E$, then $G = (V,E)$ is said to be a directed graph. A labeled graph has labels associated with its edges and vertices.

In knowledge discovery systems using a graph-based approach, the data mining algorithm uses graphs as the knowledge representation; this means that the data preparation phase includes the transformation of the data to a graph format. The *search space* of a graph-based data mining algorithm consists of all the subgraphs that can be derived from its input graph.

In the literature there exist several definitions about what a spatial data is. However, before to present our spatial model definition, we precise the following issues:

- When we speak about geometric object we refer to an object describing a form (i.e. point, line, and polygon).

- A spatial object refers to an object represented by a coordinate system (i.e. Cartesian coordinates).
- A geographic object maybe considered a specialization of a spatial object because it is represented by a coordinate system but related to earthly coordinates (sometimes called Geodetic or Geographic coordinates).

A model is a simplification of the reality. It is not the reality, rather it represents the reality. So, what a model is used for is to explain or to understand the reality. A model can be, for instance, an equation, a hypothesis or a structured idea. A spatial model is therefore an abstraction of spatial data to generate useful information to help us understand, describe, and predict how things work and/or solve problems in the real world. When we work with geographic coordinates we can talk about a geographic model.

3.2 Methodology

The idea is to create a graph-based model to represent together spatial data, non-spatial data and the spatial relations between spatial objects. We will generate datasets composed of graphs with a set of these three elements. We argue that by mining a dataset with these characteristics a data mining algorithm can search patterns involving all these elements at the same time improving the results of the spatial analysis process.

For example, finding out interesting patterns of objects located at some distance from a particular point; focusing in a current problem such as finding risk zones near the

Popocatépetl volcano (México). In addition, it would be important to know the characteristics of the evacuation routes which would be used in situations of volcanic activity, i.e., what are the characteristics of the evacuation routes; could they withstand the atmospheric conditions and the passage of vehicles in an emergency situation?

A significant characteristic of spatial data is that the attributes of the neighbors of an object may have an influence on the object itself. So, we propose to include in the model the three types of relationship mentioned in chapter 2: topological, distance, and direction relations.

As we mentioned, the basis is that in a spatial database there exist spatial objects, these objects interact with other objects (spatial relations) and they may have several attributes describing them (non-spatial data). Thus, we propose to create graphs that help us to describe these interconnections between all these elements.

A simple way can be as follows: for each object we create a vertex representing the object itself and join two vertices with a directed labeled edge if they have a spatial relation in common (we add one edge for each spatial relation among vertices). Additionally, we can also create vertices representing the value of their non-spatial attributes and directed labeled edges joining each attribute to its object (one edge per attribute).

But there is a higher complexity level when working with general graphs (i.e. a graph with multiple edges between vertices) than with simple graphs (i.e. a graph with at most one edge between any given pair of vertices and with no loops); therefore, our idea is to work

with simple graphs. So, we propose to improve this representation to the one shown in Figure 3.1 (using *UML* notation).

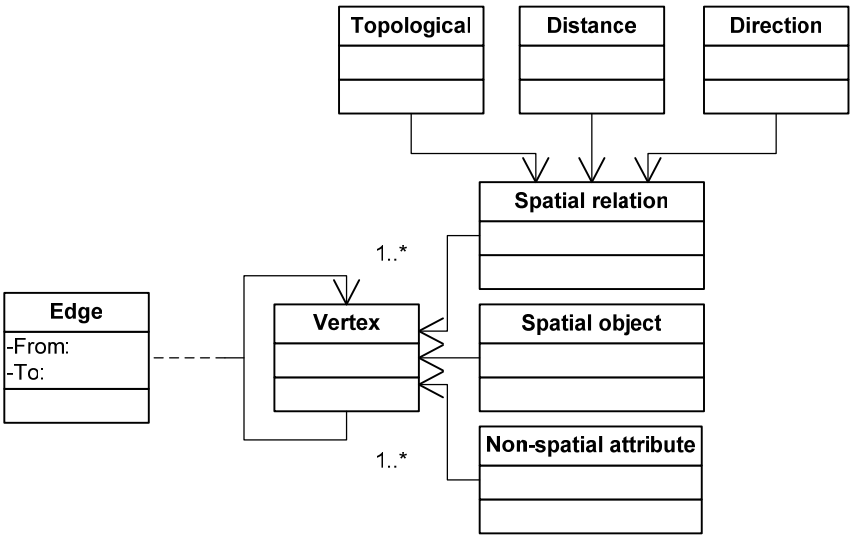


Figure 3.1. General graph-based model to represent spatial data.

In the model, the spatial data (i.e. spatial objects), non-spatial data (i.e. attributes), and spatial relations are represented as a collection of one or more directed graphs. Therefore, a directed graph contains a collection of vertices and edges representing all these elements.

Vertices represent either spatial objects, spatial relations between two spatial objects (binary relation), or non-spatial attributes describing the spatial objects. Edges represent a link between two vertices of any type. According to the type of vertices that an edge joins, an edge can represent either an attribute name or a spatial relation name. The attribute name can refer to a spatial object or a non-spatial entity. We use directed edges to represent directional information of relations among elements (i.e. object *x* touches object *y*) and to describe attributes about objects (i.e. object *x* has attribute *z*).

This knowledge representation has the capability to describe a spatial dataset using graphs, allowing a graph-based mining tool to mine it as a whole. The capabilities of the model to represent the relation between these objects will be of great impact in the results of the data mining processes since the world is described by objects and the relation between these objects, we can figure out the relations as the elements describing the interaction of the objects with each other.

3.3 Spatial Graph-based Data Representations

In the construction of the graph there are issues such as the graph complexity and size that have a direct impact over the data mining algorithm performance. The quality of results refers to another important aspect. Testing several representations will allow us to produce comparisons among obtained results, to evaluate them, and finally to make a decision for selecting the one(s) which offers the better results.

For better results we refer to our criteria for success. One approach is to show that the model allows discovering known patterns. Another approach is to have a domain expert saying that the discovered patterns are interesting. We have worked with domain experts for developing these tasks.

Currently, we have developed five models to represent spatial data, non-spatial data and spatial relationships among the spatial objects as a unique dataset from the general model.

Three issues define the characteristics (i.e. number of vertices and edges, simple graph, etc.) of the graphs created from these models:

- Equivalent spatial relations.
- Symmetric spatial relations.
- The way to represent objects and their relations in the model.

In the following subsections we will explain how these three characteristics affect the structure and composition of the graph. First, we will talk about the equivalent relations, next the symmetric relations and finally the five created models.

Equivalent Spatial Relations

Suppose that our dataset is composed of an object *A* disjoint of an object *B*, this implies that object *B* is disjoint of object *A*. In this case the two objects are disjoint each other (an equivalent relation). When creating the graph, this relation can be represented by two directed edges, one edge labeled as "DISJOINT" from object *A* to object *B* and vice versa. However we can use the following principle:

2 directed edges, 1 edge $e(v_i,v_j)$ and 1 edge $e(v_j,v_i)$	equal to	1 undirected edge $e = ij$

By applying this principle we can replace the two directed edges by only one undirected edge labeled as the equivalent relation without losing the representation of the spatial relation among the objects.

The equivalent relations implemented in our research are the following:

- TOUCH
- OVERLAPBDYINTERSECT
- OVERLAPBDYDISJOINT
- EQUAL
- CLOSE

Symmetric Spatial Relations

Suppose that our dataset is composed of an object *A* South of an object *B*, when we create the graph this relation is represented by a directed edge from object *A* to object *B* labeled as "South_of". But it implies the relation object *B* is North of object *A* (it is a symmetric relation). This last relation in some models is not represented.

According to the model the representation of a symmetric relation implies one of the following options:

- The addition of a directed edge labeled as the symmetric relation.
- The addition of a vertex labeled as the spatial relation (i.e. topological, direction) and a directed edge labeled as the symmetric relation.

The symmetric relations implemented in our research are the following:

- CONTAINS ←→ INSIDE
- COVERS ←→ COVEREDBY
- ON ←→ COVERS

- NORTH_OF ←→ SOUTH_OF
- EAST_OF ←→ WEST_OF
- NORTHEAST_OF ←→ SOUTHWEST_OF
- SOUTHEAST_OF ←→ NORTHWEST_OF

Models

As we have commented, testing several representations will allow us to produce comparisons among obtained results, to evaluate them, and finally to make a decision for selecting the one(s) which offers the better results (in term of quality). The following five models were developed to answer issues such as: in some models we obtain a reduction in the number of vertices and edges, but do they give us the same representation of our data? Are we gaining in the size reduction, but what are we loosing? There is a higher complexity working with general graphs than with simple graphs, does it affect the generated results? What about time processing?

In order to descriptive the characteristics of each model we will use the sample dataset shown in Figure 3.2. Our dataset is composed of two spatial objects, object *A* representing a *house* and object *B* representing a *lake*, and the following three spatial relations among them:

- Distance relation
    - Object *A* close object *B* (equivalent relation).
- Topological relation
    - Object *A* touch object *B* (equivalent relation).

- Direction relation
    - Object *A* South of object *B* (symmetric relation).

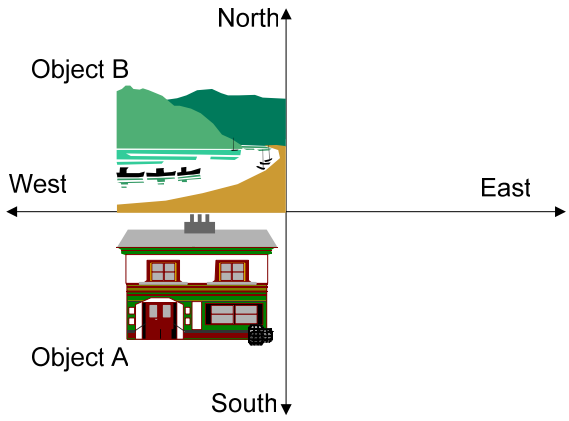


Figure 3.2. Sample dataset.

Additionally, to evaluate the characteristics of each model (the model is itself a graph) we have developed the following nine evaluation metrics:

- **Num. vertices.** Total number of vertices in the graph.
- **Num edges.** Total number of edges in the graph.
- **Size (vertices + edges).** Total number of vertices plus total number of edges in the graph.
- **% increment.** This item represents the percent of increment (in term of vertices plus edges) of this graph respect to the graph created by using the base model.
- **Simple graph.** To indicate if the graph is a simple one (graph with at most one edge between any given pair of vertices and with no loops).
- **Directed edge.** To indicate if the graph has directed edges.
- **Undirected edge.** To indicate if the graph has undirected edges.

- **Complete information.** This item shows if the symmetric relation is represented in the graph.
- **Redundant "Relation" edge.** In some models we add a "Relation" edge for representing explicitly the fact there is a relation between two spatial objects. There is a way for avoiding creating complex graphs. This item tells us if a redundant "Relation" edge is presented in the graph.

Model #1 - base model

Figure 3.3 shows the first model created for representing spatial data as proposed. The characteristics of the model according to the metrics are:

- **Num. vertices:** 2 vertices for representing each spatial object (i.e. object *A*, and object *B*).
- **Num. edges:** 4 edges, 3 edges for representing the original relations (i.e. "close", "touch", and "South_of" relations) and 1 edge for representing the "North_of" relation created from the original "South_of" symmetric relations. The "North_of" relation is itself a symmetric relation.
- **Size (vertices + edges):** 6
- **% increment:** 0%, it is the *base model*.
- **Simple graph.** No, it is a complex graphs with 4 edges linking 2 vertices.
- **Directed edge.** Yes, they are used for representing the "South_of" and "North_of" symmetric relations. The direction of the edges is according to the lecture of the relations among the objects.

- **Undirected edge.** Yes, they are used for representing the "close" and "touch" equivalent relations.
- **Complete information.** Yes, in the graph are represented the "North_of" symmetric relation created from the original "South_of" symmetric relation.
- **Redundant "Relation" edge.** No, in the model we don't use "Relation" edges.

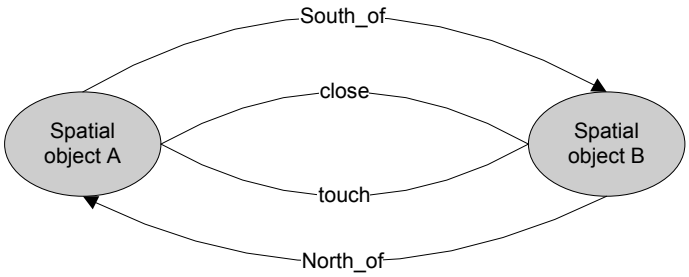


Figure 3.3. Model #1 - base model.

Model #2 - single replication of relations types, complete information

In Figure 3.4 we show our second model created for representing spatial data. The characteristics of the model according to the metrics are:

- **Num. vertices:** 5 vertices, 2 vertices for representing the spatial objects and 3 vertices for representing the "topological", "distance", and "direction" relations (the spatial relation types). For each spatial relation type among two spatial objects we add 1 vertex labeled as its name. In the example there exist 1 "topological" relation, 1 "distance" relation, and 1 "direction" relation.
- **Num. edges:** 6 edges, 3 edges for representing the original relations, 2 edges for representing the equivalent relations (i.e. "close" and "touch" relations) created from the original ones, and 1 edge for representing the symmetric relation (i.e. "North_of" relation) created also from the original relations.

- **Size (vertices + edges):** 11
- **% increment:** +83.33%
- **Simple graph.** Yes, there exists at most 1 edge between any given pair of vertices.
- **Directed edge.** Yes, they are used for representing all relations. The direction of the edges is from the vertices representing the spatial objects to the vertices representing the spatial relation types.
- **Undirected edge.** No, in the model we don't use undirected edges.
- **Complete information.** Yes, we represent the symmetric relations created from the original ones.
- **Redundant "Relation" edge.** No, in the model we don't use "Relation" edges.

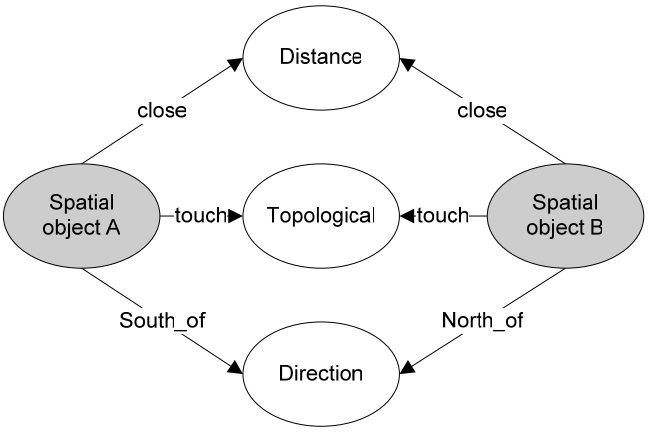


Figure 3.4. Model #2 - single replication of relations types, complete information.

Model #3 - double replication of relations types, no complete information

In Figure 3.5 we show our third model created for representing spatial data. The characteristics of the model according to the metrics are:

- **Num. vertices:** 8 vertices, 2 vertices for representing the spatial objects and 6 vertices for representing the "distance", "topological", and "direction" relations. For

each spatial relation type among two spatial objects we add 2 vertices labeled as its name. For instance, in the example there exist three relations: 1 "topological" relation, 1 "distance" relation, and 1 "direction" relation, so we add 6 vertices, 2 per each spatial relation.

- **Num. edges:** 9 edges, 6 edges (the "Relation" edges) to link the vertices representing the spatial objects to the vertices representing the spatial relation types (from each vertex representing a spatial object start 3 edges since we have 3 relations), and 3 edges for representing the original relations. These last 3 edges are used to join the vertices representing the spatial relation types.
- **Size (vertices + edges):** 17
- **% increment:** +183.33%
- **Simple graph.** Yes, there exists at most 1 edge between any given pair of vertices.
- **Directed edge.** Yes, they are used for representing the symmetric relations and the "Relation" edges. The direction of the "Relation" edges is from the vertices representing the spatial objects to the vertices representing the spatial relation types. The direction of the other edges is according to the lecture of the relations among the objects.
- **Undirected edge.** Yes, they are used for representing the equivalent relations.
- **Complete information.** No, we don't represent the symmetric relations created from the original ones.
- **Redundant "Relation" edge.** Yes, we use in the model "Relation" edges for representing explicitly the existence and type of a relation among 2 spatial objects. Additionally, we use this edge for avoiding creating complex graphs.

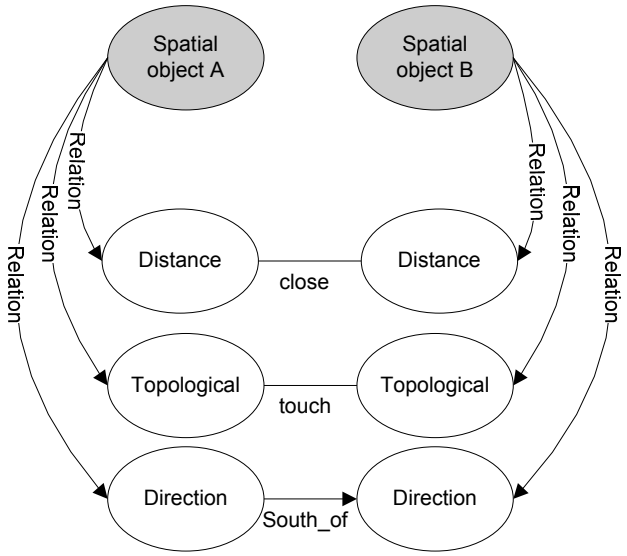


Figure 3.5. Model #3 - double replication of relations types, no complete information.

Model #4 - single replication of relations types, no complete information

In Figure 3.6 we show our fourth model created for representing spatial data. The characteristics of the model according to the metrics are:

- **Num. vertices:** 5 vertices, 2 vertices for representing the spatial objects and 3 vertices for representing the "topological", "distance", and "direction" relations. For each spatial relation type among two spatial objects we add 1 vertex labeled as its name. In the example there exist 1 "topological" relation, 1 "distance" relation, and 1 "direction" relation.
- **Num. edges:** 6 edges, 3 edges (the "Relation" edges) to link a vertex representing a spatial object (in the example we have 2 vertices since there are 2 spatial objects) to the vertices representing the spatial relation types (from this vertex representing a spatial object start 3 edges since we have 3 relations), and 3 edges for representing

the original relations. These last 3 edges are used to link the vertices representing the spatial relation types to the un-used vertex representing the other spatial object.

- **Size (vertices + edges):** 11
- **% increment:** +183.33%
- **Simple graph.** Yes, there exists at most 1 edge between any given pair of vertices.
- **Directed edge.** Yes, they are used for representing the symmetric relations and "Relation" edges. The direction of the "Relation" edges is from a vertex representing a spatial object to the vertices representing the spatial relation types. The direction of the other edges is from the vertices representing the spatial relation types to the un-used vertex representing the other spatial object (it is according to the lecture of the relations among the objects).
- **Undirected edge.** Yes, they are used for representing the equivalent relations.
- **Complete information.** No, we don't represent the symmetric relations created from the original ones.
- **Redundant "Relation" edge.** Yes, we use in the model "Relation" edges for representing explicitly the existence and type of a relation among 2 spatial objects. Additionally, we use this edge for avoiding creating complex graphs.

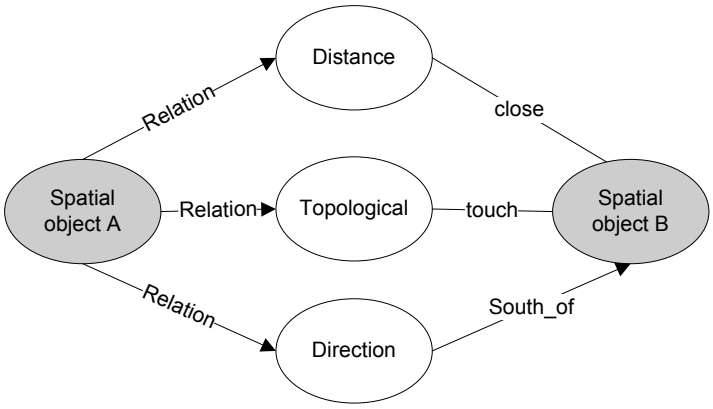


Figure 3.6. Model #4 - single replication of relations types, no complete information.

Model #5 - double replication of relations types, complete information

In Figure 3.7 we show our fifth model created for representing spatial data. The characteristics of the model according to the metrics are:

- **Num. vertices:** 8 vertices, 2 vertices for representing the spatial objects and 6 vertices for representing the "distance", "topological", and "direction" relations. For each spatial relation type we add 2 vertices labeled as its name. In the example we add 2 vertices for the 2 "topological" relations, 2 vertices for the "distance" relation, and 2 vertices for the "direction" relation.
- **Num. edges:** 12 edges, 6 edges (the "Relation" edges) to link the vertices representing the spatial objects to the vertices representing the spatial relation types, and 6 edges for representing the original relations and those ones generated from them (equivalent and symmetric relations). These last 6 edges are used to link the vertices representing the spatial relation types to the vertices representing each spatial object (3 edges for each spatial object).
- **Size (vertices + edges):** 20

- **% increment:** +233.33%
- **Simple graph.** Yes, there exists at most 1 edge between any given pair of vertices.
- **Directed edge.** Yes, they are used for representing all relations.
- **Undirected edge.** No, in the model we don't use undirected edges.
- **Complete information.** Yes, we represent the symmetric relations created from the original ones.
- **Redundant "Relation" edge.** Yes, we use in the model "Relation" edges for representing explicitly the existence and type of a relation among 2 objects. Additionally, we use this edge for avoiding creating complex graphs.

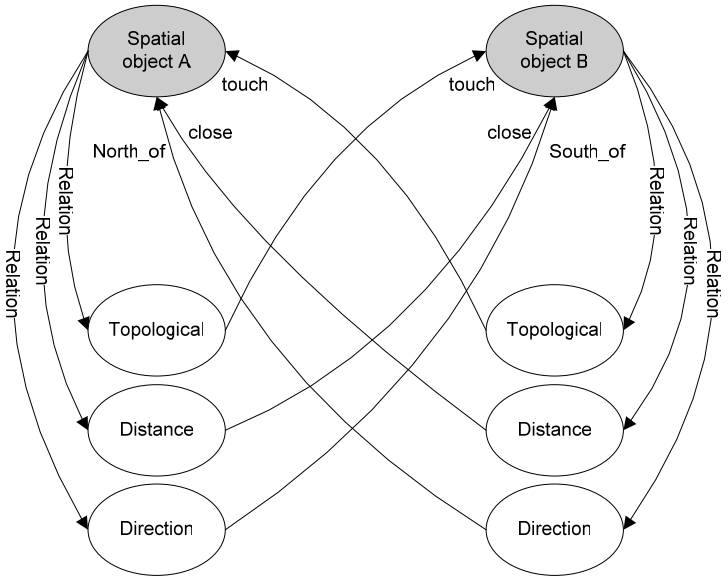


Figure 3.7. Model #5 - double replication of relations types, complete information.

Table 3.1 presents the results of the nine metrics developed to evaluate the characteristics of each model. Model #1 is named the base model.

Model	Num. Vertices (1)	Num. Edges (2)	Size (v + e) (3)	% Increment (4)	Simple Graph (5)	Directed Edge (6)	Undirected Edge (7)	Complete Information (8)	Redundant "Relation" Edge (9)
#1	2	4	6	-	No	Yes	Yes	Yes	No
#2	5	6	11	+83.33	Yes	Yes	No	Yes	No
#3	8	9	17	+183.33	Yes	Yes	Yes	No	Yes
#4	5	6	11	+83.33	Yes	Yes	Yes	No	Yes
#5	8	12	20	+233.33	Yes	Yes	No	Yes	Yes

Table 3.1. Characteristics of the graph-based representation models.

The metrics were proposed based on the causes/effects each of them has both into the created graph and the mining algorithm. We visualize four significant issues related directly to these metrics:

1. Search space

The search space of a graph-based data mining algorithm consists of all the subgraphs that can be derived from its input graph, thus, the number of vertices (1) and edges (2) of the created graph (3) define the size of the search space for the discovery system. Therefore, the objective must be to minimize the number of vertices and edges used to create the graphs but at the same time to maximize the representativeness of the dataset. As we can see in Table 3.1, the model using the minimum number of vertices and edges to represent our sample dataset is model #1 (2 vertices and 4 edges) whereas model #5 is the opposite case (8 vertices and 12 edges).

2. Processing time

The search space size plays a relevant role regarding to the processing time used to discover patterns. If we have a large search space the algorithm would require more time to evaluate all the possible subgraphs. Therefore, a comparison among the "percentage of increment" (4) metric of the proposed models is presented in Table 3.1. Remember this metric compares the size of a given model with respect to model #1 (base model). For instance, model #5 has a graph size increment of 233.33% respect to model #1. In other words, the mining algorithm will require the evaluation of 233.33% more vertices and/or edges by using model #5 instead of model #1 for the same dataset.

3. Graph Complexity

Next chapter describes the Subdue system, our graph-based data mining tool. As we will see there exists a higher complexity for the mining algorithm to work with complex graphs than with simple ones (i.e. at most an edge among any given pair of vertices and with no loops). For instance, into the graph match process, expanding phase (Subdue uses an "expanding" approach for discovering patterns), and graph compression stage. Therefore, the objective was to propose graph-based models that allow us to create simple graphs. As we can see in Table 3.1, only model #1 does not create simple graphs.

Thus, as a strategy to break the multiplicity of edges among two vertices (for instance, vertices representing two spatial objects meeting two or more spatial relations among them) we use the following approaches:

- To add a new vertex labeled as the relationship type (i.e. Topological, Distance, and Direction) by each spatial relation among the spatial objects. This approach is used in model #2.
- To add a new vertex (model #4) or two new vertices (model #3 and model #5) labeled as the relationship type (i.e. Topological, Distance, and Direction) by each spatial relation among the spatial objects, and to link this new vertex (model #4) or new vertices (model #3 and model #5) with the vertices representing the spatial objects by using edges labeled as "Relation". The approach used to link the vertices is different in each model as we have mentioned in the definition of each of them. This nomenclature is used to represent the fact there exits a spatial relation among these spatial objects. These edges are known as redundant relation edges (9).

4. Data representativeness

The directed edges (6), undirected edges (7), and complete information (8) metrics are used to maximize the representativeness of the dataset but minimizing, as most as possible, the graph size and its complexity. Directed edges are used to represent the symmetric spatial relations (object *A* North_of object *B,* implies, *B* South_of *A*), the redundant relation edges, and the non-spatial attributes describing the spatial objects. Undirected edges are used to represent equivalent spatial relations (the relation is represented by an undirected edge instead of two directed edges). Finally, complete information means that symmetric spatial relations among spatial objects are also represented into the model.

3.4 Use-case

To show the applicability of our model, we will use a dataset from a Popocatépetl volcano database [37][40]. The database contains data related to several issues in the zone such as settlements, rivers, and evacuation roads in the zone, just to mention some of them.

Figure 3.8 shows a fragment of the layers “roads” and “settlements” of the Popocatépetl volcano zone. The roads layer (shown in green color) represents the roads in the zone; it is composed of spatial objects (i.e. lines) and non-spatial data describing the characteristics of those roads (i.e. id, start point, end point, length, and type). The settlements layer (shown in pink color) represents population areas in the zone. The layer is composed of spatial objects (i.e. polygons) and non-spatial data describing the characteristics of the settlements (i.e. id, area, perimeter, and type).

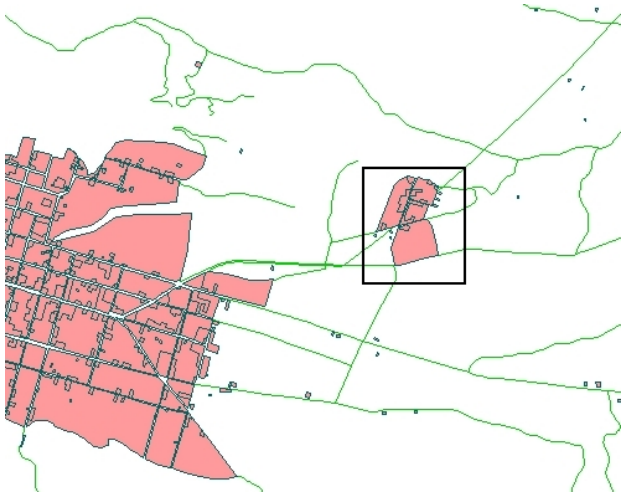


Figure 3.8. Spatial database representing some objects of the world.

Suppose we are interested in the identification of relations among the roads starting in, or crossing a settlement (i.e. type and a set of characteristics of roads in relation to the number of people living in a settlement that in case of a volcanic contingency are required to be evacuated). So, we need to create a dataset involving these elements for mining it and search for patterns that help us to evaluate the characteristics of the roads and, may be, to make the decision of improving them (or build new ones) for their utilization in case of volcano activity. In this case we are working with different types of spatial objects and also we are adding non-spatial data and relationships (as touch or overlap) to our dataset.

But the construction of the dataset implies being concerned about some constraints. For example, if we include all the elements existing in the data layers we might build a huge graph, and this will have a direct impact in the data mining algorithm. So, we explore methodologies to deal with topics such as complexity, the size of the graph, noise and quality of the data. A solution for facing the problem of creating a huge graph is delimiting the set of elements to be included in it by using *selection windows*. The user can create these windows and only the elements inside them are candidates to be included as objects in the graph. The idea is shown in Figure 3.9. Suppose the user will work with *n* data layers, thus, for each layer we will select only the spatial objects inside the area delimited by the *selection window*. We can see this functionality such as a drill operation over all the spatial layers the user works with.

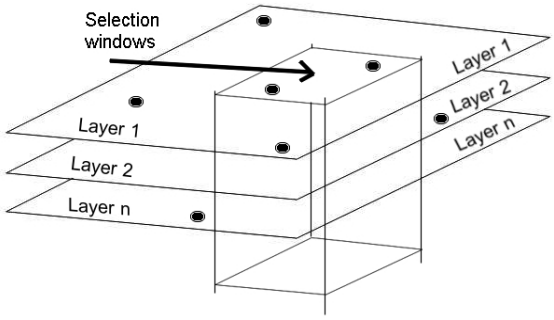


Figure 3.9. Selection window.

Once we have defined the working area, as shown in Figure 3.8, we can build the graph. The process consists of three phases. In the first phase, the user has to choose the spatial relation(s) to evaluate among the spatial objects (in the example the touch or overlap topological relations). The second phase involves the validation process, only the objects covered by the relation(s) become elements to be included in the graph.

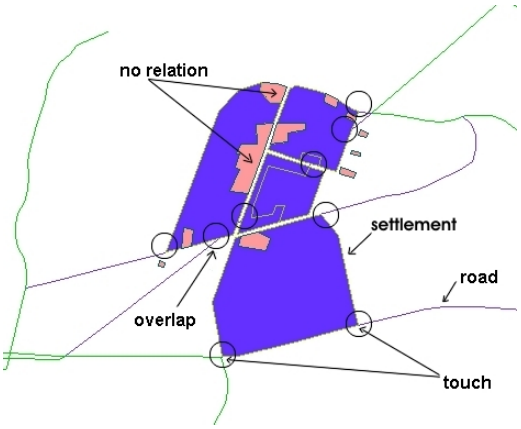


Figure 3.10. Querying a spatial database.

The last phase consists of building the graph using the results of the validation process. Figure 3.10 presents an example of this functionality. This time only the area delimited by a *selection window* is shown. In the figure, the spatial objects setting either the touch or

overlap relations are shown in blue color, the rest of the objects are shown in their original colors. The circles show some examples where a road is starting or crossing a settlement.

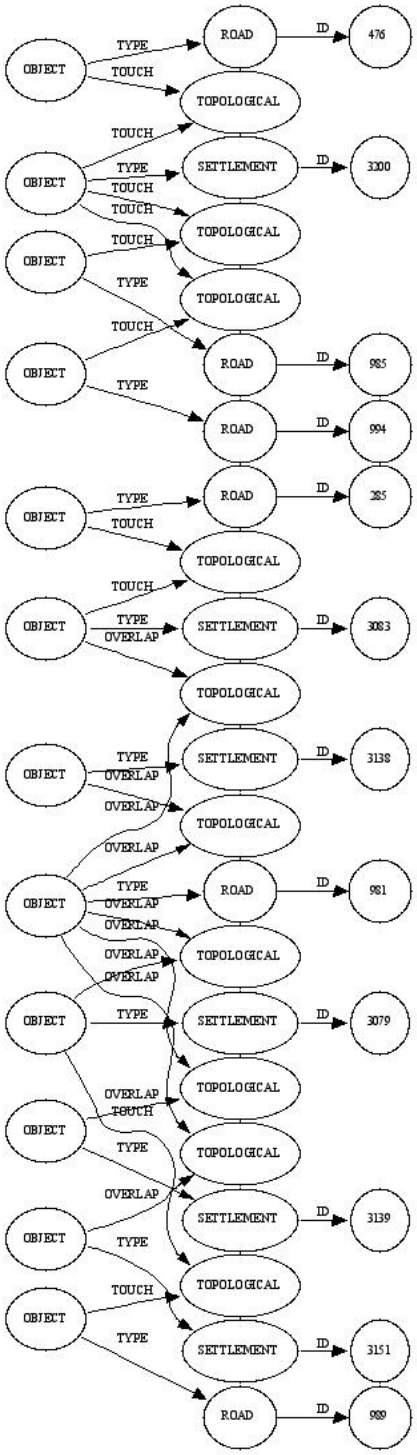


Figure 3.11. Graph-based representation for spatial data.

Figure 3.11 shows the graph created using model #2 (we used the Graphviz System [19] to draw the graph). The vertices in the graph represent either spatial objects (i.e. road, settlement), spatial relations between two objects (i.e. topological), or attributes describing the objects (i.e. ID value). In this example, we use a special vertex labeled as "object" for expressing the interconnection between the spatial objects, their spatial relations, and non-spatial attributes (i.e. object type road with ID 989 overlaps or touches with object…).

According to the type of vertices that an edge joins, the edge can be labeled as either a spatial relation name (i.e. overlap, touch), or as the name of an attribute describing the characteristics of an object (i.e. type, ID).

We may read the graph as follows: there are six roads and six settlements inside the working area meeting either a touch or overlap relation in the form road → settlement. We suppose that each polygon object in the map represents a settlement. Some roads start from a settlement and others cross a settlement. For example, the road with ID *981* crosses the settlements with ID's *3079*, *3139*, *3151*, *3083* and *3138*. This means that we need to be careful with this road since it has interaction with several settlements and in case of a contingency it will be used widely. In the graph we included only the object's ID non-spatial attribute for each object.

This is a simple example; in the real world the spatial data layers may have hundreds or thousands of objects, and each object may have dozens of attributes describing it. Joining all these elements will allow creating large graphs representing the elements found in a

spatial database improving the results of the data mining task. Once we have created the graph, it will be used as input to a graph-based mining system.

3.5 Conclusion

Our idea is to propose a graph-based model to represent together spatial data, non-spatial data and the spatial relations between spatial objects. Based in the model we generate datasets composed of graphs with a set of these three elements. Our argumentation is that by mining a dataset with these characteristics a data mining algorithm can search patterns involving all these elements at the same time improving the results of the spatial analysis process.

We have presented an example of the applicability of the proposal using data from a Popocatépetl volcano database. The created graph will be the input for a graph-based mining system. We propose to use the Subdue system, which will be described in the next chapter.

Chapter 4

MINING THE GRAPH

Chapter 4 describes the characteristics of the Subdue system [23][43], our data mining tool. Subdue was developed at the University of Texas at Arlington and it can be applied to any domain that can be represented as a graph. In the first part of the chapter we present the general characteristics and main functions. In the second part we describe the overlap feature and its importance for the pattern discovery system, and introduce a new algorithm named limited overlap.

4.1 Characteristics

The Subdue system discovers substructures using a graph-based representation of structural databases. Structural data involves the concept of units or parts and the interdependence and relationships of those parts. The substructures (a connected subgraph within the graphical representation) describe concepts in the data (i.e. patterns).

The substructure discovery system represents structural data as a labeled graph. The input to Subdue is a graph where labeled vertices correspond to objects in the data, and directed or undirected labeled edges map relationships between objects.

The discovery algorithm is a computationally constrained beam search. There are three different evaluation methods to guide the search towards more appropriate substructures: Minimum Description Length (*MDL*), Size-based, and Set Cover. The default evaluation method is *MDL*.

The algorithm begins with the substructure matching a single vertex in the graph. Each iteration the algorithm selects the best substructure and incrementally expands the instances of the substructure. An instance of a substructure in the input graph is a subgraph that matches (graph theoretically) that substructure.

The algorithm searches for the best substructure until all possible substructures have been considered or the total amount of computation exceeds a given limit. Evaluation of each substructure is determined by how well the substructure compresses the description length of the input graph.

There might be slight variations of some substructures that can be considered as instances of another substructure. Subdue uses an inexact graph match algorithm to identify this kind of instances. In this inexact match approach, each distortion of a graph is assigned a cost and if the total cost is lower than a given threshold, the substructure is considered an instance of the other substructure.

A distortion is described in terms of transformations such as the deletion, insertion, and substitution of vertices and edges. The best substructure found by Subdue can be used to compress the input graph, which can then be input to another iteration of Subdue. After several iterations, Subdue builds a hierarchical description of the input data where later substructures are defined in terms of substructures discovered on previous iterations.

Figures 4.1, 4.2, 4.3 and 4.4 show an example of the system's functionality. The example is presented in terms of the house domain, where a house is defined as a triangle on a square. *T* represents a *triangle*, *S* a *square*, *C* a *circle* and *R* a *rectangle* (see Figure 4.1). The objects in the figure (i.e. *T1*, *S1*, *R1*) become labeled vertices in the graph, and the relationships (i.e. *on*(*S1*, *R1*), *shape*(*C1*, *circle*)) become labeled edges.

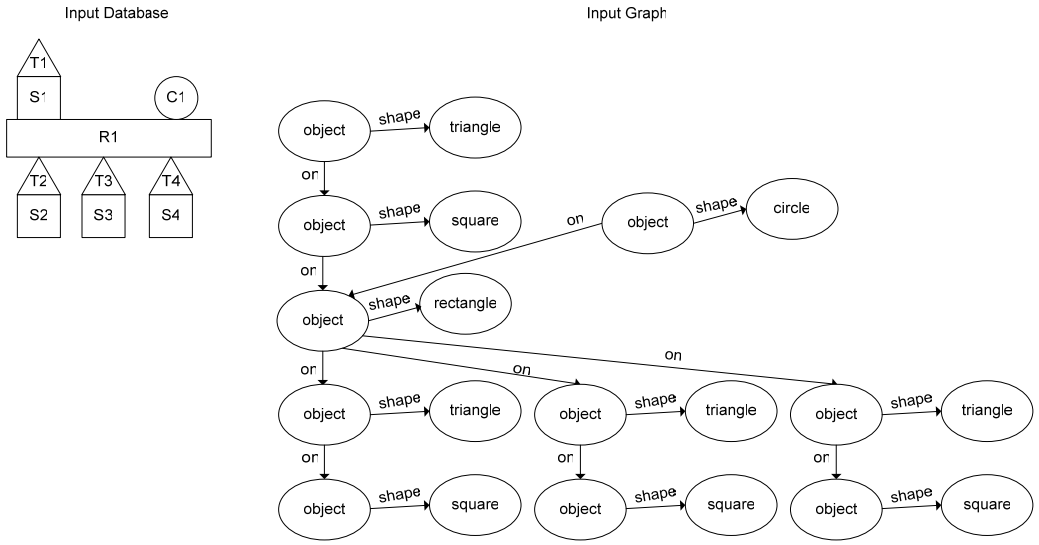


Figure 4.1. Graph representation of the house domain.

The graph representation of the substructure discovered by Subdue from this data is shown in Figure 4.2 where Subdue found four instances of "*triangle on square*".

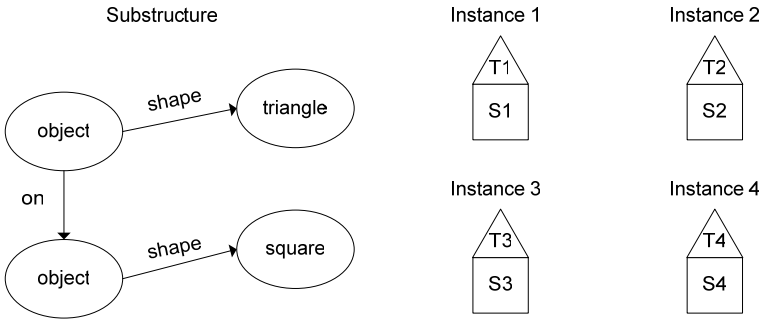


Figure 4.2. Substructure and instances discovered from the house domain by Subdue.

After a substructure is discovered, each instance of the substructure in the input graph is replaced by a single vertex representing the entire substructure. In Figure 4.3 the discovered substructure (*object shape triangle on object shape square*) is labeled as *SUB_1* (**SUB**structure number 1).

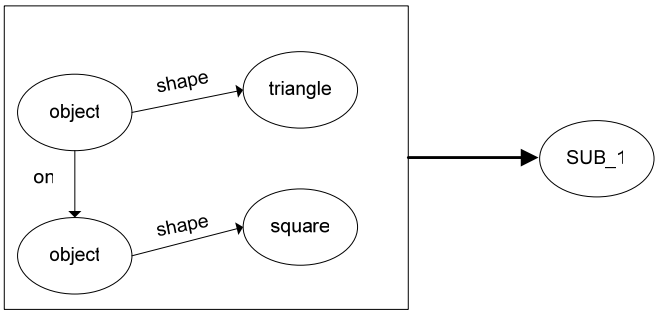


Figure 4.3. Substructure replacement procedure in the house domain.

Finally, the substructure (labeled as *SUB_1*) is used to compress the original input graph, which can then be input to another iteration of Subdue (see Figure 4.4).

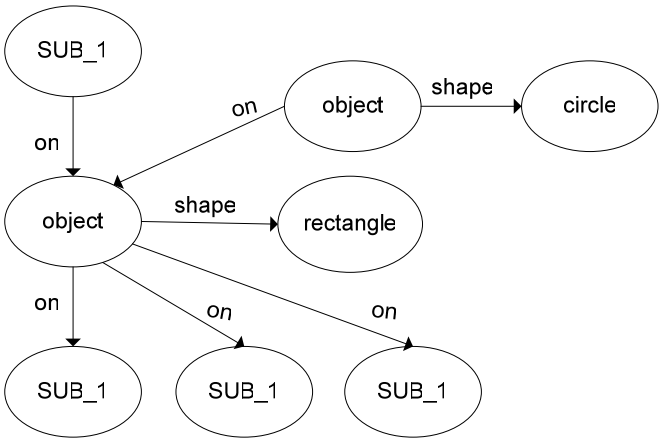


Figure 4.4. Graph representation of the house domain after substructure replacement.

The Subdue system's ability to perform discovery has been proved in several domains including scene analysis, chemical analysis and *CAD* circuit analysis. In [18] the authors present an example of the Subdue capabilities to perform data mining tasks, they work with an earthquake database and show that Subdue is capable of finding not only the shared characteristics of the earthquake events, but also space relations between them. In the case of the identification of shared characteristics, they used the pattern containing the region number specification to recognize the area being studied; in the case of space relations, they found patterns that represent parts of the paths of the involved fault (i.e. subarea with a high concentration of earthquakes).

4.1.1 Main Functions

In this subsection we present a briefly description of the main functions composing the core of Subdue. Each of them is itself integrated by several subfunctions, but the idea is to present them in a global perspective.

Compress

The compress function returns a new graph, which is the given graph compressed with the given substructure instances. Vertices "*SUB*" replace each instance of the substructure, and "*OVERLAP*" edges are added between vertices "*SUB*" of overlapping instances. Edges connecting to overlapping vertices are duplicated, one per each instance involved in the overlap.

Discover

This function plays the role of manager in the phase of discovering the best substructures in an input graph. It is in charge of issues such as to get initial substructures, to extend each substructure, to evaluate each extension, and to add to a final list the best discovered substructures.

Evaluate

This function implements the different evaluation methods used to guide the search towards more appropriate substructures: Minimum Description Length (*MDL*), Size-based, and Set Cover. The value of a substructure *s* in a graph *g* is computed as:

$$size(g) / (size(s) + size(g|s))$$

The value of *size*() depends on whether we are using the *MDL* or Size-based evaluation method. If *MDL* is used, then *size*(*g*) computes the description length in bits of *g*. In case of Size-based, then *size*(*g*) is simply *vertices*(*g*) + *edges*(*g*). The *size*(*g*|*s*) is the size of graph *g* after compressing it with substructure *s*. Compression involves replacing each

instance of *s* in *g* with a new single vertex and reconnecting edges external to the instance to point to the new vertex. For Size-based, the *size*(*g*|*s*) can be computed without actually performing the compression.

If negative graphs are present, then the evaluation becomes:

$$\frac{size(Gp) + size(Gn)}{size(S) + size(Gp|S) + size(Gn) - size(Gn|S)}$$

where *Gp* is the positive graph and *Gn* is the negative graph.

If the evaluation method is Set Cover, then the evaluation of substructure *s* becomes:

$$\frac{(\textit{num positive edges containing } s) + (\textit{num negative edges not containing } s)}{(\textit{num positive edges}) + (\textit{num negative edges})}$$

Extend

This function returns a list of substructures representing extensions to the given substructure. Extensions are constructed by adding an edge (or edge and new vertex) to each positive instance of the given substructure in all possible ways according to the graph. Matching extended instances are collected into new extended substructures, and all such extended substructures are returned. If a negative graph is present, then instances of the substructure in the negative graph are also collected.

Graphmatch

The Graphmatch function returns true if *graph_1* and *graph_2* match with cost less than the given threshold. If so, the objective is to store the match cost in the variable pointed to by

matchCost and to store the mapping between *graph_1* and *graph_2* in the given array is non null.

Subgraph Isomorphism

Set of functions implementing a subgraph isomorphism algorithm. The objective is to find predefined substructures: searches for subgraphs of *graph_2* that match *graph_1* and returns the list of such subgraphs as instances in *graph_2*. Returns empty list if no matches exist. This procedure mimics the "discover substructures" loop by repeatedly expanding instances of subgraphs of *graph_1* in *graph_2* until matches are found.

4.2 Overlap

Now, we will talk about the overlap feature in Subdue. This feature has a preponderant role in the substructure discovery system. The overlap feature is controlled by the overlap user's parameter. If overlap is false then overlap among instances is not allowed, otherwise, overlap is allowed.

To explain the cause/effect of this feature in Subdue, we will present some examples generated by using two standalone functions belonging to the Subdue: the Subgraph Isomorphism and Minimum Description Length functions. The results generated by these functions are affected by the value of the overlap parameter.

Subgraph Isomorphism (SGISO)

As we have already mentioned, the subgraph isomorphism function searches for subgraphs of *graph_2* that match *graph_1* and returns the list of such subgraphs as instances in *graph_2*. The subgraph isomorphism function is embodied in the "FindInstances" function. The procedure is optimized toward *graph_1* being a small graph, and *graph_2* being a large graph.

The FindInstances function starts finding a list of single-vertex instances, one for each vertex in *graph_2* that matches the first vertex of *graph_1*. Next, the algorithm attempts to extend each instance in the instance list by an edge (or edge and new vertex) from *graph_2* that matches the attributes of the given edge in *graph_1*. Finally, the instances not matching *graph_1* are filtered, and an overlapped instances validation process is preformed. If overlap is false overlapped instances are discarded. The resulting list represents the instances of *graph_1* in *graph_2*.

For illustrative purpose suppose we have as input graphs those shown in Figure 4.5 and Figure 4.6. Input *graph_1* has 3 vertices and 2 edges, and *graph_2* has 8 vertices and 7 edges.

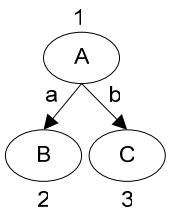


Figure 4.5. SGISO - input graph_1.

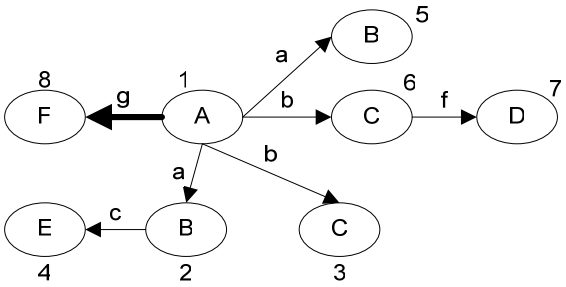


Figure 4.6. SGISO - input graph_2.

Running *SGISO* with the overlap parameter set to false, the algorithm finds 1 instance of *graph_1* in *graph_2* (see Figure 4.7). Figure 4.8 shows the discovered instance in *graph_2*.

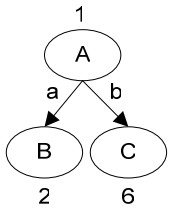


Figure 4.7. SGISO - no overlap.

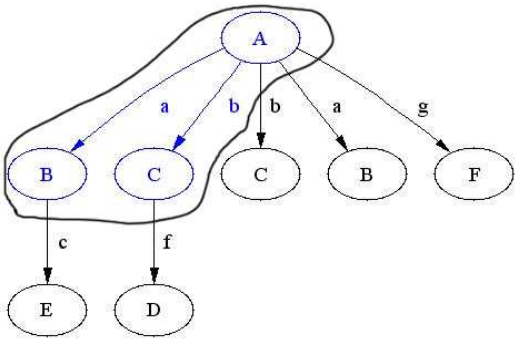


Figure 4.8. SGISO - no overlap, 1 instance in graph_2.

Now, running *SGISO* with the overlap parameter set to true, the algorithm finds the 4 instances shown in Figure 4.9. We can see that each instance has the vertices labeled as *A*, *B*, *C* but they represent different vertices (they have different *ID* vertices). For example, the first instance has the *ID* vertices 1, 5 and 3; the second instance has the *ID* vertices 1, 5 and 6 respectively, and so on. Figure 4.10 shows the 4 discovered instances in *graph_2*.

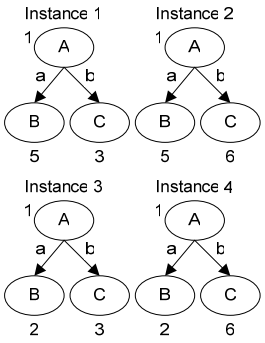


Figure 4.9. SGISO - overlap.

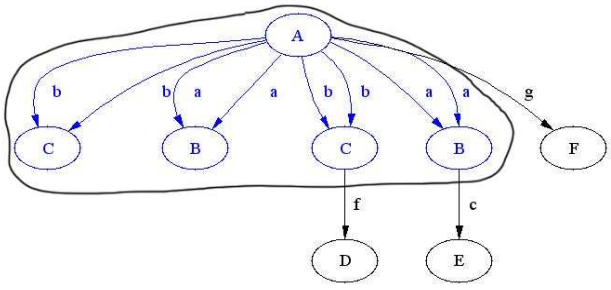


Figure 4.10. SGISO - overlap, 4 instances in graph_2.

Minimum Description Length (MDL)

The *MDL* principle states that the best theory to describe a set of data is that theory which minimizes the description length of the entire dataset. We define the *MDL* of a graph to be the number of bits necessary to completely describe the graph. The minimal encoding of the graph in terms of bits is computed as follows:

$$MDL = vertexBits + rowBits + edgeBits$$

Where *vertexBits* represents the number of bits needed to encode the vertex labels of the graph, *rowBits* represents the number of bits needed to encode the rows of the adjacency matrix *A* (the matrix represents the graph connectivity), and *edgeBits* represents the number of bits needed to encode the edges represented by the entries $A[i,j] = 1$ of the adjacency matrix *A*.

The standalone *MDL* function computes the description length (*dL*) of *graph_1*, *graph_2* and *graph_2* compressed with *graph_1* along with the final *MDL* compression measure:

$$dL(graph\_B)/dL(graph\_A)+dL(graph\_B|graph\_A)$$

dL(*graph_2*|*graph_1*) represents the value of *graph_2* compressed with *graph_1*. The overlap feature has a direct effect to compute this value because the number of instances for compressing *graph_2* is based in the instances list returned by the FindInstances function (see example standalone subgraph isomorphism function).

As we have commented, the Subdue system implements a compress graphs function named CompressGraph. The inputs of the function are the graph to be compressed, and the

substructure instances used to compress the graph. The function returns a new graph, which is the given graph compressed with the given substructure instances.

In order to show the effects of the overlap in the standalone *MDL* function (focusing in the generated compressed graph) we will show, in the following figures, examples of the different cases of validation that are implement to face this feature. Figure 4.11 shows the input *graph_1*, this graph will be used in all the examples as the first input graph. It has 3 vertices and 2 edges.

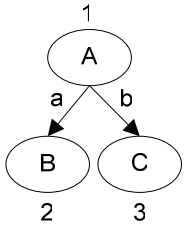


Figure 4.11. MDL - input graph_1.

In our first example we will use as *graph_2* (our second graph) the shown in Figure 4.12. This graph has 8 vertices and 7 edges. As we can see our graph has a "remarked" edge, the directed edge labeled as *g* between vertex 1 (labeled as *A*) and vertex 8 (labeled as *F*). This edge represents the current case of validation (in the future it will be named the *guide edge*).

As part of the CompressGraph algorithm, there is a function named AddOverlapEdges that adds edges to the compressed graph, describing overlapping instances of the substructure in the given graph, based in two conditions. First, if two instances overlap, then an undirected

edge *OVERLAP* is added between them. Second, if an external edge points to a vertex shared between multiple instances, then duplicate edges are added to all instances sharing the vertex. If the second condition is true the AddOverlapEdges function calls a new function named AddDuplicateEdges. The purpose of this function is to add duplicate edges based on overlapping vertex between substructures.

As we can see, the layout of the edge (that we call the *guide edge*) is important for the global functionality of the compress graph algorithm. So, in the following examples we will change the guide edge for illustrating the different validation cases.

Returning to our example, if we run the *MDL* program with the overlap parameter set to false, our final compressed graph will be the shown in Figure 4.13. Since overlap is not allowed between instances, the FindInstances function finds 1 instance of *graph_1* that match *graph_2*. In the graph this instance was replaced by a vertex *SUB*, so we have only 1 vertex of this type. There are not edges *OVERLAP*, and no edges were duplicated.

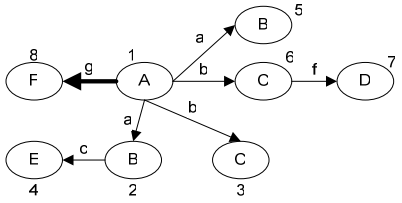


Figure 4.12. MDL example 1 - input graph_2.

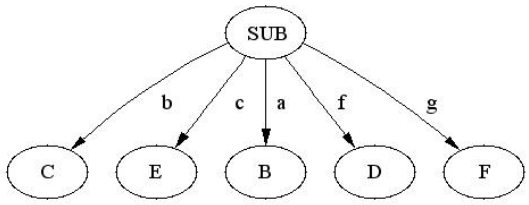


Figure 4.13. MDL example 1 - no overlap.

In the next example our *graph_2* is the shown in Figure 4.14. It is the same graph used in the previous example, but this time we run *MDL* with the overlap parameter set to true. The generated compressed graph is shown in Figure 4.15. The FindInstances function finds 4

instances since the overlap between instances is allowed. In the graph the 4 instances were replaced by 4 vertices *SUB*. There are 5 edges *OVERLAP*, an edge *OVERLAP* between two vertices *SUB* tells us that these substructures overlap. By "substructures overlap or overlapped substructures" we refer that the instances they represent have common vertices.

In the example, there is a directed edge (edge *g*, our *guide edge*) connecting two vertices, one of them belonging to a substructure (in exact term, to an instance of a substructure) and the other one no belonging to the substructure (vertex *F*), is an external edge; the discovered instances were 4 (overlap is allowed), so, in the graph there are 4 directed edges starting from each vertex *SUB* (the direction of the edge is preserved) to the vertex *F*.

A similar procedure was implemented for edge *f* (connecting vertex *C* and vertex *D*, vertices number 6 and 7 respectively) and edge *c* (connecting vertex *B* and vertex *E*, vertices number 2 and 4, respectively). They were duplicated since these edges connect vertices belonging to a substructure to vertices no belonging to the substructure (external edges). In the figure we can see that vertex *D* has 2 edges *f* connecting it to 2 vertices *SUB* (preserving the direction of the edge) and vertex *E* has 2 edges *c* connecting it to 2 vertices *SUB* (remember that 4 instances were discovered, overlap is allowed).

Finally, our compressed graph has 7 vertices: 4 vertices *SUB*, 1 vertex *F*, 1 vertex *D*, and 1 vertex *E*; and 13 edges: 5 edges *OVERLAP*, 4 edges labeled as *g*, 2 edges labeled as *f*, and 2 edges labeled as *c*.

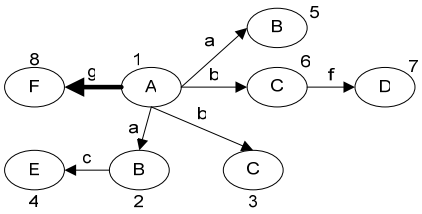


Figure 4.14. MDL example 2 - input graph_2.

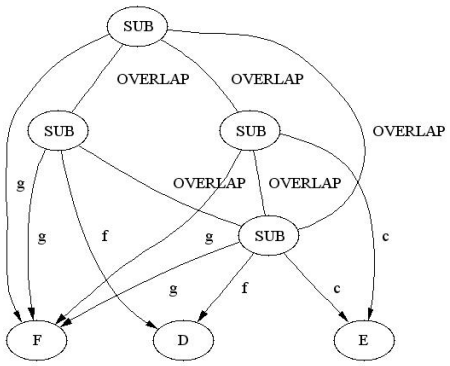


Figure 4.15. MDL example 2 - overlap.

For the following examples the overlap parameter is always set to true.

Figure 4.16 shows our new *graph_2*, we only changed the direction of the *guide edge*. Now it starts from vertex *F* (number 8) to vertex *A* (number 1). The generated compressed graph is shown in figure 4.17. In the compressed graph we observe that, this time, the 4 edges *g* start from vertex *F* to the 4 vertices *SUB* (the direction of the edge is preserved). Everything else remains as the previous example.

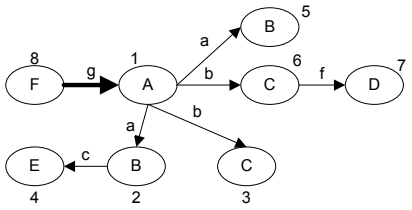


Figure 4.16. MDL example 3 - input graph_2.

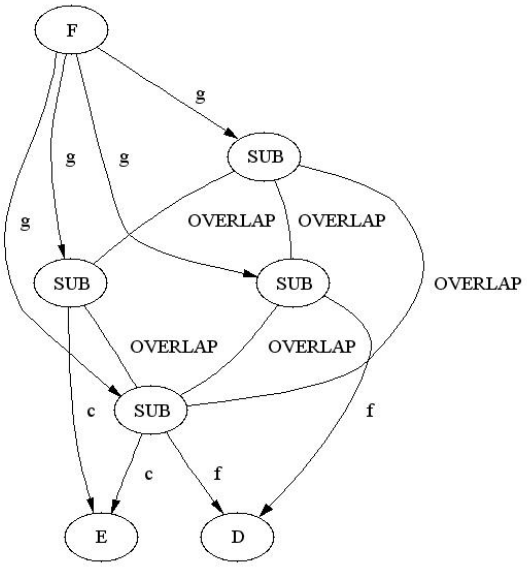


Figure 4.17. MDL example 3 - overlap.

For the next example, we only change from a directed *guide edge* to an undirected one. Our *graph_2* is the graph shown in Figure 4.18. By running *MDL* we generate the compressed graph presented in Figure 4.19. The only modification is that the edges between vertex *F* and the 4 vertices *SUB* are undirected edges. Everything else remains unchanged.

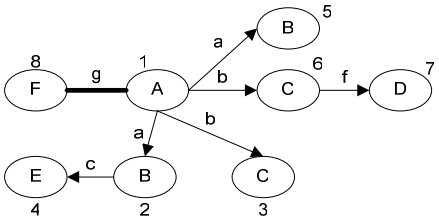


Figure 4.18. MDL example 4 - input graph_2.

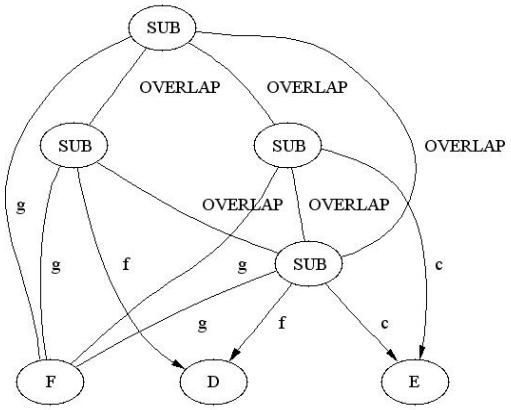


Figure 4.19. MDL example 4 - overlap.

For illustrating a new outcome of the overlap feature in the standalone *MDL* function, we change our *graph_2* as follows:

- Vertex number 8 labeled as *F* is deleted.
- Our *guide edge* labeled as *g* is deleted.
- A new edge labeled as *g* (our *guide edge*) is added between vertex number 1 labeled as *A* and vertex number 3 labeled as *C*. For the current example we use a directed edge.

Our *graph_2* is shown in Figure 4.20. The graph has 7 vertices and 7 edges. The "new" edge has the characteristic that it is an edge between two vertices belonging to a same substructure and in some cases, as we will see, they belong to overlapped substructures. Remember that overlap is allowed between instances, so the FindInstances function finds 4 instances.

The generated compressed graph is shown in Figure 4.21. We mentioned that our *guide edge* joins 2 vertices belonging to a same substructure (vertex number 1 labeled as *A* and vertex 3 labeled as *C*, and in some cases these vertices belong to overlapped substructures. So, in the graph there are 6 edges labeled as *g*, 4 edges joining vertices *SUB* (telling us that they overlap), and two self edges in 2 vertices *SUB*. These last 2 edges are our new case of validation. The self edge tells us that inside the substructure, there is an edge joining two vertices belonging to the same substructure. The direction of the edges does not matter

since this is an edge inside the substructure (an internal edge). The other duplicated edges (i.e. edge *c* and edge *f*) were computed using the previous described procedure.

Finally, our compressed graph has 6 vertices: 4 vertices *SUB*, 1 vertex *D*, and 1 vertex *E*; and 15 edges: 5 edges *OVERLAP*, 6 edges labeled as *g*, 2 edges labeled as *f*, and 2 edges labeled as *c*.

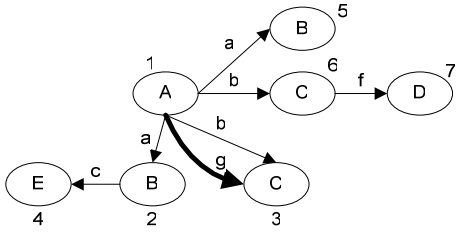


Figure 4.20. MDL example 5 - input graph_2.

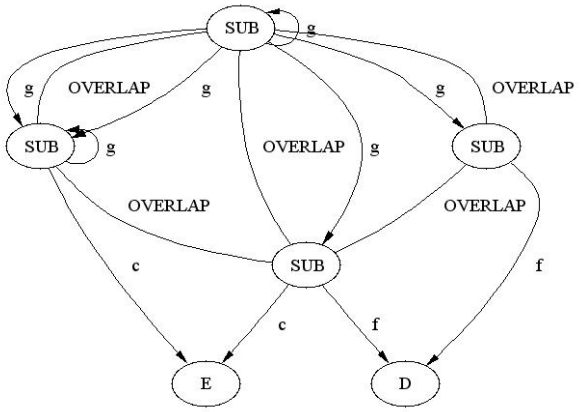


Figure 4.21. MDL example 5 - overlap.

For the example shown in Figure 4.22, we only changed the direction of the *guide edge*. Now it starts from vertex number 3 labeled as *C* to vertex number 1 labeled as *A*. The generated compressed graph is presented in Figure 4.23. The generated compressed graph is the same one that in the previous example. This is telling us that just changing the direction of the *guide edge* does not have effect in the resulting compressed graph.

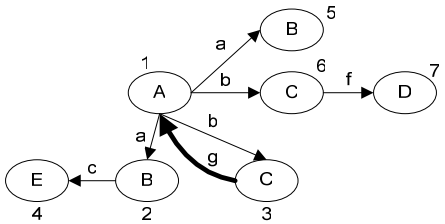


Figure 4.22. MDL example 6 - input graph_2.

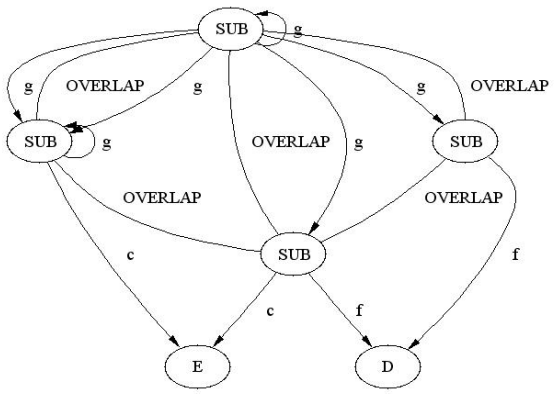


Figure 4.23. MDL example 6 - overlap.

For the next example, we only change from a directed *guide edge* to an undirected one. Our *graph_2* is the graph shown in Figure 4.24. The generated compressed graph is presented in Figure 4.25. The single change is that all edges *g* are now undirected edges. Everything else remains unchanged.

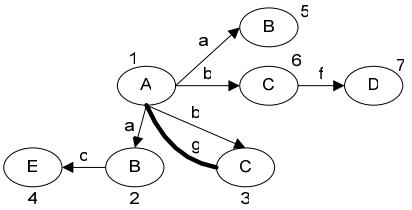


Figure 4.24. MDL example 7 - input graph_2.

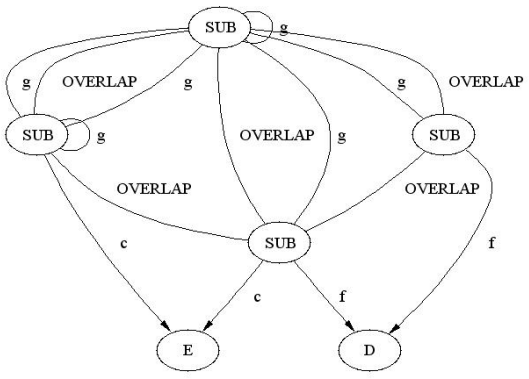


Figure 4.25. MDL example 7 - overlap.

Now, for illustrating a new outcome of the overlap feature in the standalone *MDL* function (self edge in a vertex shared by instances of a substructure), we change our *graph_2* as follows:

- Edge labeled as *g* between vertex number 1 labeled as *A* and vertex 3 labeled as *C* is deleted.
- A new edge labeled as *g* (our *guide edge*) is added between vertex number 1 labeled as *A* and vertex number 1 labeled as *A* (a self edge). For the current example a directed edge.

Our *graph_2* is shown in Figure 4.26. The graph has 7 vertices and 7 edges. The "new" edge has the characteristic that is a self edge; it starts and ends in the same vertex. This vertex belongs to overlapped substructures, so we have 4 substructures sharing it.

The resulting compressed graph is shown in Figure 4.27. Since our *guide edge* is a self edge the compressed graph has 10 edges labeled as *g*. There are 6 edges joining vertices *SUB* (telling us that they are overlapped substructures), and 4 self edges in 4 vertices *SUB* telling us that they have an edge which origin and destination vertices are the same.

We note in the compressed graph a special characteristic between some vertices *SUB*. We refer that some pairs of vertices *SUB* are joined by 2 edges labeled as *g*. One of them starts in the first vertex *SUB* and ends in the second one. The second edge has a vice versa direction. This is consequence that they are overlapped substructures with a common vertex and this vertex is the source and destination of an internal edge.

Finally, our compressed graph has 6 vertices: 4 vertices *SUB*, 1 vertex *D*, and 1 vertex *E*; and 19 edges: 5 edges *OVERLAP*, 10 edges labeled as *g*, 2 edges labeled as *f*, and 2 edges labeled as *c*.

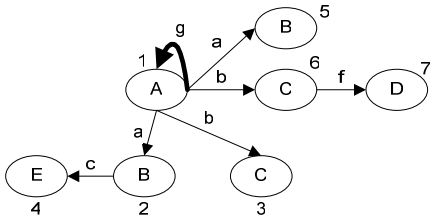


Figure 4.26. MDL example 8 - input graph_2.

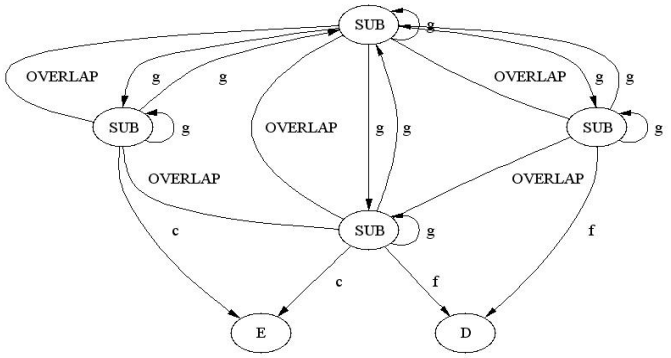


Figure 4.27. MDL example 8 - overlap.

For the last example, we use as *graph_2* the one shown in Figure 4.28. We only change our *guide edge* from a directed edge to an undirected one. Figure 4.29 shows the generated compressed graph. There are 2 differences between this compressed graph and the previous one. First, the edges labeled as *g* are undirected edges. Second, the special characteristic for some vertices *SUB* is implemented in a different way.

Currently, we note that those vertices *SUB* joined by 2 edges labeled as *g* in the previous example, now, they are joined by just 1 undirected edge labeled as *g*. Since the *guide edge* is undirected we only need 1 edge for representing they are overlapped substructures with a common vertex and this vertex is the source and destination of an internal undirected edge. The 4 vertices *SUB* have self edges, but they are undirected ones this time. Everything else remains unchanged.

Finally, our compressed graph has 6 vertices: 4 vertices *SUB*, 1 vertex *D*, and 1 vertex *E*; and 16 edges: 5 edges *OVERLAP*, 7 edges labeled as *g*, 2 edges labeled as *f*, and 2 edges labeled as *c*.

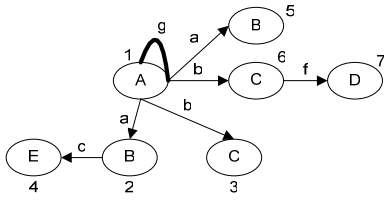


Figure 4.28. MDL example 9 - input graph_2.

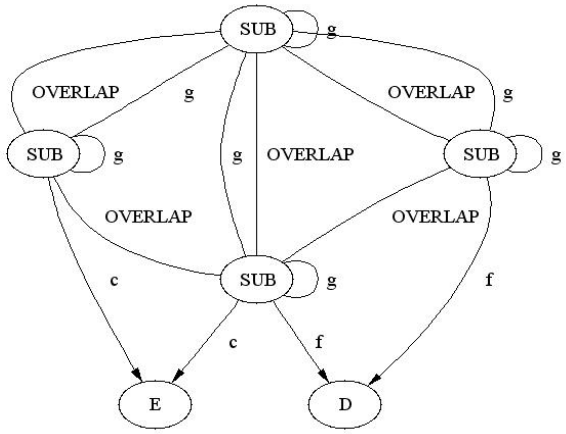


Figure 4.29. MDL example 9 - overlap.

4.3 Limited Overlap

As we have described in the previous section, the overlap feature plays a preponderant role in the Subdue's substructures discovery system. As consequence, the generated results are also conditioned, in a high percentage, to this parameter. However, as we have seen, it is implemented to allow overlap among any instances of a substructure or among all the instances of a substructure. In this context, we propose a new approach named limited overlap. The major feature in this approach is to give the user the means to specify the set of vertices where an overlap is allowed. These vertices may represent significant elements in the context we work with.

In the following subsections we will present the motivation, advantages, new algorithm, and an example of the new overlap feature implemented in the Subdue system.

Motivation

As we have already commented, the current overlap feature in Subdue is implemented in an orthodox way: all or nothing. It means that Subdue allows the overlap among all the instances sharing at least one vertex or that Subdue does not allow (discard) the overlap among instances sharing at least one vertex.

But we argue that a third option is needed, an option where the user has the capability to set over which vertices the overlap will be allowed, it is a limited overlap. We visualize directly three motivations issues to propose the implementation of the new algorithm that will be explained in the following subsections:

- Search space reduction.
- Processing time reduction.
- Specialized overlapping pattern oriented search.

In order to help us to describe the characteristics of the limited overlap we will use the graphs show in Figure 4.30 and 4.31 as input graphs in the future examples. Input graph *PS_1* (**P**redefined **S**ubstructure number 1) has 2 vertices and 1 edge, input graph *PS_2* (**P**redefined **S**ubstructure number 2) has also 2 vertices and 1 edge, and finally *graph_3* has 9 vertices and 8 edges.

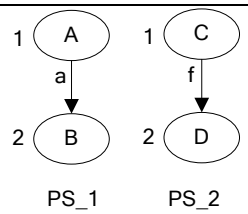


Figure 4.30. Limited overlap - input graphs PS_1 and PS_2.

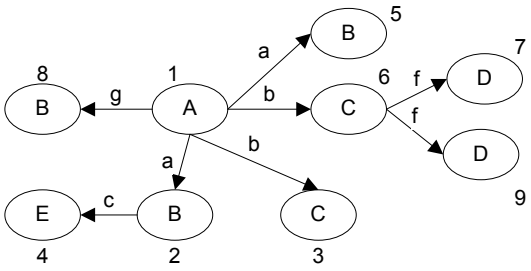


Figure 4.31. Limited overlap - input graph_3.

1) Search space reduction. In knowledge discovery systems using a graph-based approach, the data mining algorithm uses graphs as a knowledge representation; the search space of a graph-based data mining algorithm consists of all the subgraphs that can be derived from its input graph.

The substructures discovery process in Subdue begins with the creation of the substructures matching a single vertex in the graph (one for each of the different labels in the graph). Each iteration through the algorithm selects the best substructures and expands the instances of these substructures by one neighboring edge (or an edge and new vertex) in all possible ways.

But as part of the process to select the best substructures and then to expand them there exists a filter phase. In this phase according to the overlap parameter the instances of a substructure are evaluated: if overlap is set to true then overlapped instances are kept, otherwise is overlap is set to false then overlapped instances are discarded.

For example, suppose we want to search the instances of *PS_1* and *PS_2* in *graph_3*. With the overlap set to false Subdue discovers 2 instances, one instance for each predefined substructure as we can see en Figure 4.32. *PS_SUB_X* is the nomenclature used by Subdue to identify it is an instance of Predefined Substructure "SUB_X"; where SUB_X means it is the SUBstructure number X. But, if overlap is set to true, Subdue discovers 4 instances, 2 instances for each predefined substructure (see Figure 4.33). Finally, suppose the user wants to search instances of *PS_1* and *PS_2* in *graph_3* but this time he/she considers that vertices *A* have a higher relevance (for example, a remarkable spatial object in a spatial database) so he/she proposed to use a limited overlap, the overlap will be allowed just among instances containing vertices *A*. We can see that *PS_1* has a vertex *A*, thus this time Subdue finds 3 instances, 2 instances of *PS_1* and 1 instance of *PS_2* as we show in Figure 4.34. In the figure we can observe that instance 1 and instance 2 share the vertex number 1 labeled as *A*. For *PS_2* Subdue finds 1 instance since the other one (instance number 4 in Figure 4.33) has also the vertex number 6 labeled as *C*, but the overlap is just allowed among vertices *A*.

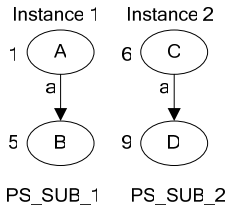


Figure 4.32. No overlap - SGISO.

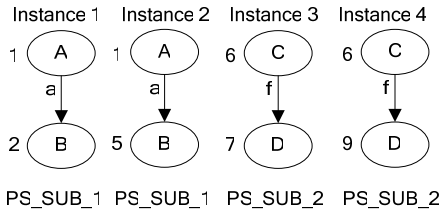


Figure 4.33. Overlap - SGISO.

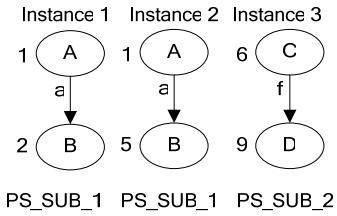


Figure 4.34. Limited overlap PS_1 - SGISO.

We have mentioned that the best discovered substructure by Subdue (by iteration) can be used to compress the input graph, which can then be input to another iteration. After several iterations, Subdue builds a hierarchical description of the input data where later substructures maybe defined in terms of substructures discovered on previous iterations. We have also comment that each iteration through the algorithm selects the best substructures and expands the instances of these substructures by one neighboring edge (or an edge and new vertex) in all possible ways. So the number of instances of the substructures defines the search space (by iteration) in the substructure discovery process.

As we can see in our example, by using the limited overlap we obtain a search space reduction (with overlap set to true), since the number of instances becoming candidates to be expanded is selected according to the allowed values given by the user.

2) Processing time reduction. The reduction in the number of instances becoming candidates to be expanded results in a search space reduction, and this effect also has a new outcome, a processing time reduction.

Allowing overlap slows Subdue considerably since the number of candidate instances to expand, to evaluate, to match, to compress, and to discover increase as we have seen. However, by the implementation of the limited overlap, the number of instances to be processed in these phases decrease resulting also in a processing time reduction in the overall substructure discovery process.

3) Specialized overlapping pattern oriented search. We have also commented that the limited overlap gives the user the capabilities to define the set of interesting elements over which the overlap will be allowed (the elements are represented as vertices in the graph according to the proposed model) so the algorithm will discard the elements (overlapped) that the user does not considerer significant.

In the example presented in Figure 4.34, the judgment of the user is that vertices *A* have a higher relevance so he/she propose to use a limited overlap, the overlap will be allowed just among instances containing vertices *A*. This consideration is a personal decision of the user according to the work context (in many cases with the support of domain expert). For instance, a remarkable element may refer to a spatial object in a spatial database or to some characteristic defining a particular topic of a dataset.

Therefore, the limited overlap gives the user the mechanics to implement a pattern oriented search. The user delimits the set of elements that will have a preponderant role in the substructure discovery process. But this characteristic gives also a new advantage, the patterns evaluation process is simplified since the set of generated results is smaller because it is focused over the user requirements.

Algorithm

As we have described, the limited overlap gives the user the capabilities to define the set of elements (vertices in a graph) where overlap among instances is allowed. The process starts reading this set of vertices which are integrated to the Subdue's parameters as a limited overlap label list. During the substructure discovery process a filter phase is performed. This phase consists to evaluate (and may be to discard), based on the overlap parameter, the list of discovered instances.

If the discovered process is composed by several iterations, after each of them, the overlap label list may be updated to integrate the new vertices where overlap is also allowed. Remember that at the end of each iteration, the best substructure found by Subdue can be used to compress the input graph, which can then be input to another iteration of Subdue. After several iterations, Subdue builds a hierarchical description of the input data where later substructures are defined in terms of substructures discovered on previous iterations.

```
Limited Overlap (instance1, instance2)
  //Global process
  while (elementsInstance1 < totalElementsInstance1) AND NOT endProcess
      //Instance1's vertex
      vertex1 = vertex from instance1

      //Instance2's vertex
      while(elementsInstance2 < totalElementsInstance2)
          vertex2 = vertex from instance2

      //Instances overlap
      if (vertex1 = vertex2){
          intancesOverlap = true
          endProcess = true
      }

      //Instances overlap, check by limited overlap
      if (instancesOverlap AND overlapLabelList NOT EMPTY){
          if (vertex1 in overlapLabelList)
              //It is a limited overlap
              limitedOverlap = true
          else
              //Process continues until all vertex1 are validated
              endProcess = false
      }

  return instancesOverlap, limitedOverlap
```

In the validation process each vertex belonging to instance1 (*vertex1*) is validated against all vertices belonging to instance2 (*vertex2*). If *vertex1* and *vertex2* are the same then the instances overlap. If the instances overlap then *vertex1* is validated against the list containing the set of vertices allowed for overlap (the overlapLabelList). If *vertex1* exists in overlapLabelList then is a limited overlap.

Example

To illustrate the functionality of the new algorithm, we will present some examples generated using a new version of Subdue implementing the limited overlap feature. The input graphs for the examples are those shown in Figure 4.30 and Figure 4.31; the idea is to

perform a pattern oriented search by discovering patterns in *graph_3* but based on *PS_1* and *PS_2* (our predefined patterns).

Subdue implements a pattern oriented search by, initially, finding all instances of *PS_1* and *PS_2* in *graph_3*. The next step is to compress *graph_3* using the found instances of each *PS_1* and *PS_2*. Figure 4.35 shows the compressed graph with the overlap parameter set to false. As we can see, Subdue found 1 instance of *PS_1* (i.e. vertex *PS_SUB_1*) and 1 instance of *PS_2* (i.e. vertex *PS_SUB_2*) since overlap among instances is not allowed (see Figure 4.32 for details).

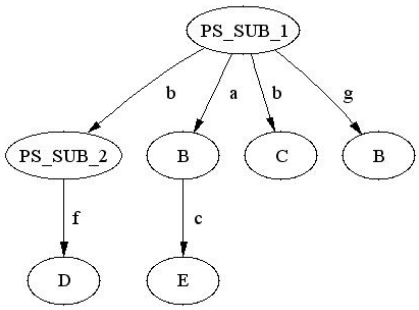


Figure 4.35. No overlap - compressed graph.

Finally, the compressed graph becomes the input graph to discover substructures. Figure 4.36 shows the best 3 substructures found by Subdue according to its substructure discovery system (see Section 4.1 for details). Each substructure has 1 instance; it means that there exists 1 repetition of each of them in the input graph (in the example we use exact graph match, but Subdue allows also inexact graph match). The first one (labeled as "a"), is composed by 2 vertices: 1 vertex *B*, and 1 vertex *E*; and 1 edge labeled as *c*. The second one (labeled as "b") is composed by 4 vertices: 1 vertex *PS_SUB_1*, 1 vertex *C*, and 2

vertices *B*; and 3 edges labeled as *a*, *b*. *g*. The vertex *PS_SUB_1* is itself a substructure composed by two vertices: 1 vertex *A*, and 1 vertex *B*; and 1 edge labeled as *a*. We have already commented that later substructures may be defined in terms of previous discovered substructures, or in term of predefined substructures as in this example. Finally, the third one (labeled as "c") is composed by 4 vertices: 1 vertex *PS_SUB_1*, 1 vertex *PS_SUB_2*, 1 vertex *C*, and 1 vertex *B*; and 3 edges: 2 edges labeled as *b*, and 1 edge labeled as *g*. Once again vertices *PS_SUB_1* and *PS_SUB_2* are themselves substructures.

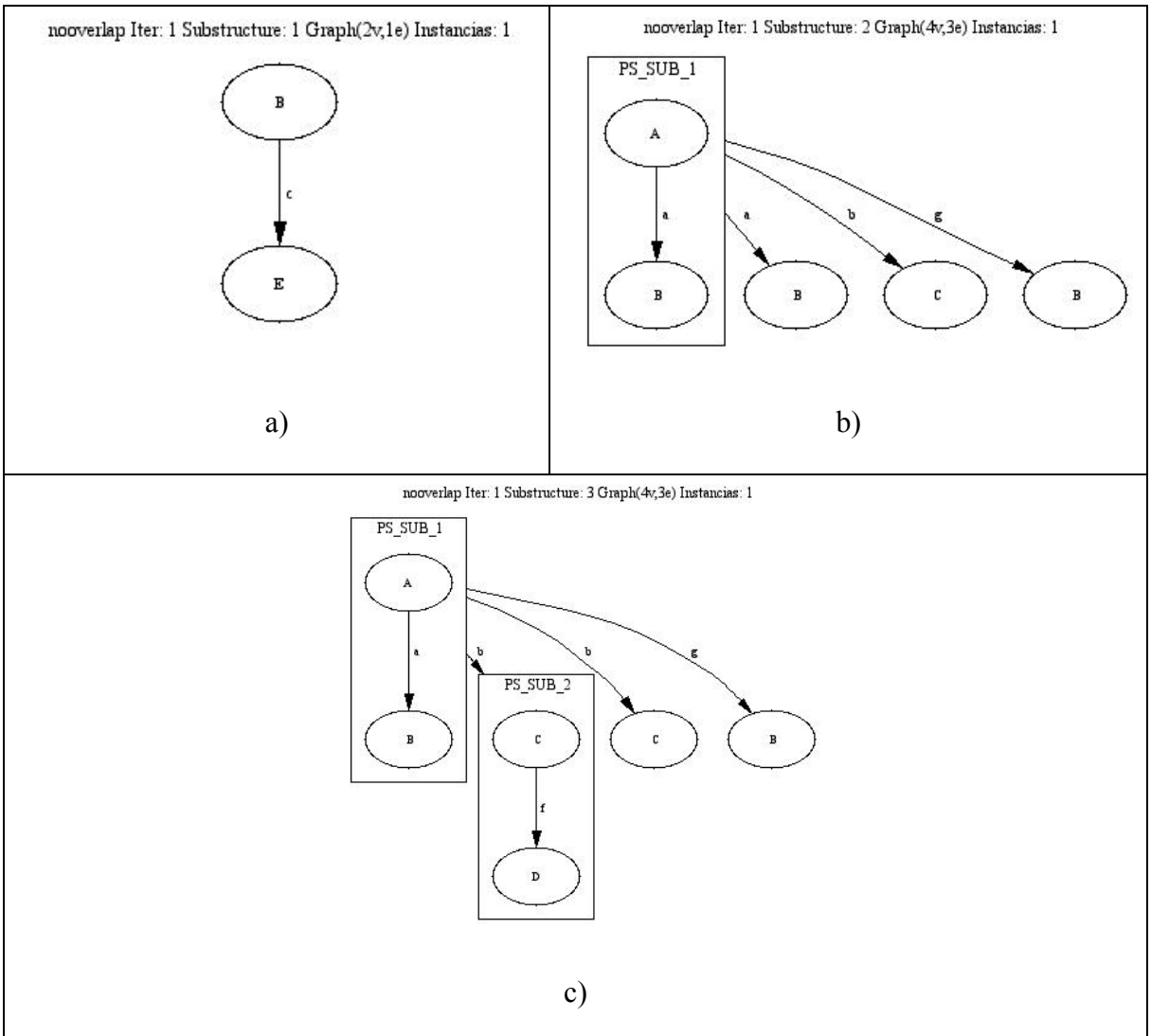


Figure 4.36. No overlap - discovered substructures.

For the next example we set overlap to true. The generated compressed graph is shown in Figure 4.37. Now, Subdue finds 2 instances for each *PS_1* and *PS_2* (see Figure 4.33 for details).

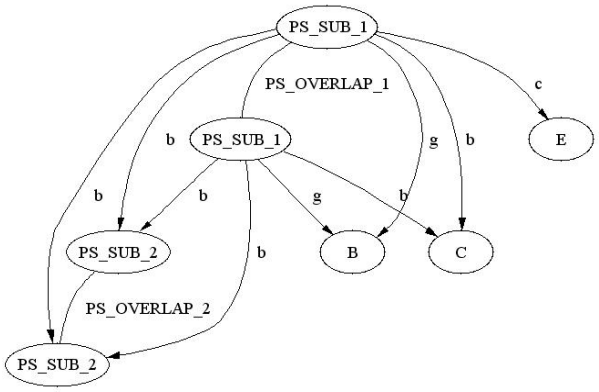


Figure 4.37. Overlap - compressed Graph.

Figure 4.38 shows the best 3 substructures discovered by Subdue. The first and second ones (labeled as "a" and "b") are themselves the predefined substructure *PS_SUB_2* and *PS_SUB_1* respectively with 2 instances each of them. The third one (labeled as "c") is a substructure composed by 2 vertices and 1 edge with 1 instance.

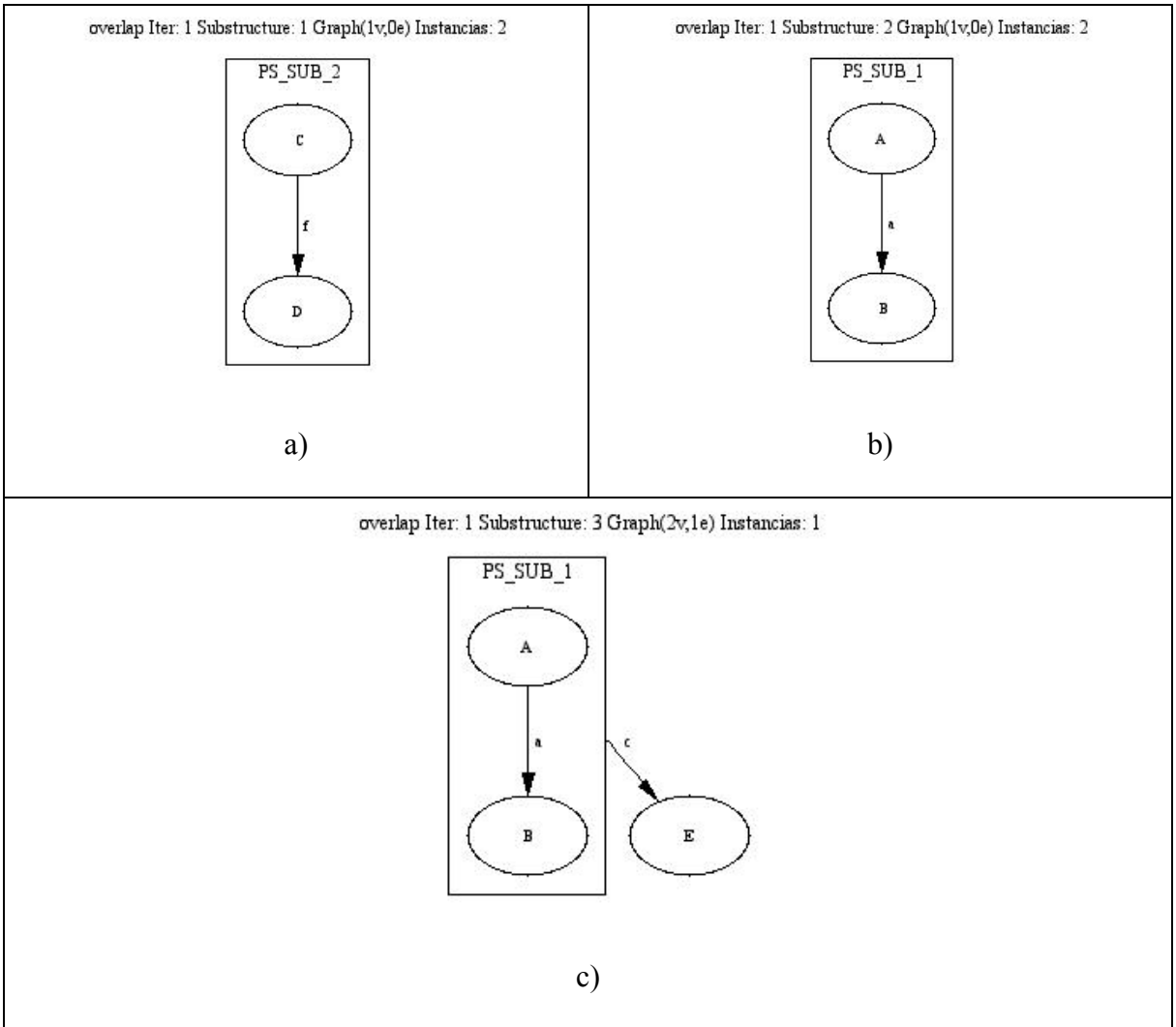


Figure 4.38. Overlap - discovered substructures.

Our next example is implemented by using the limited overlap feature. Suppose the user wants to perform a specialized overlapping pattern oriented search: he/she only wants to allow overlapped instances in vertices representing *PS_1*. The generated compressed graph is shown in Figure 4.39. As result of the restriction for overlapping instances Subdue finds 2 instances of *PS_1* (they share the allowed vertex *A*) and just 1 instance of *PS_2* (since they share a not allowed vertex *C*). The found instances are shown in Figure 4.34.

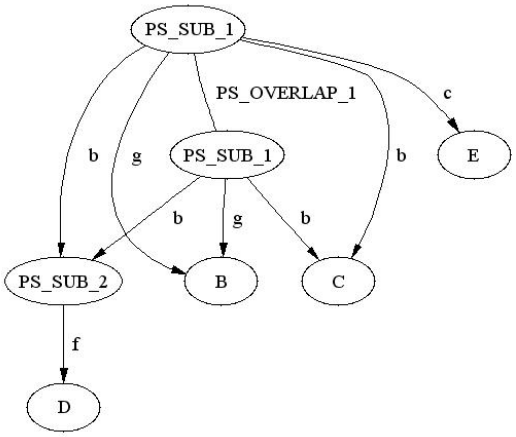


Figure 4.39. Limited overlap PS_1 - compressed graph.

Figure 4.40 shows the best 3 substructures discovered by Subdue from this graph using the limited overlap. In the figure we can see that Subdue reports as the best substructure *PS_SUB_1* with 2 instances (labeled as "a"). This is consequence of the integration of the compressed graph since it has 2 vertices *PS_SUB_1*. The second reported substructure (labeled ad "b") is composed by 2 vertices: 1 vertex *PS_SUB_1*, and 1 vertex *E*; and 1 edge labeled as *c*. The last one (labeled as "c") is composed also by two vertices *PS_SUB_1*; and 1 edge labeled as *PS_OVERLAP_1*.

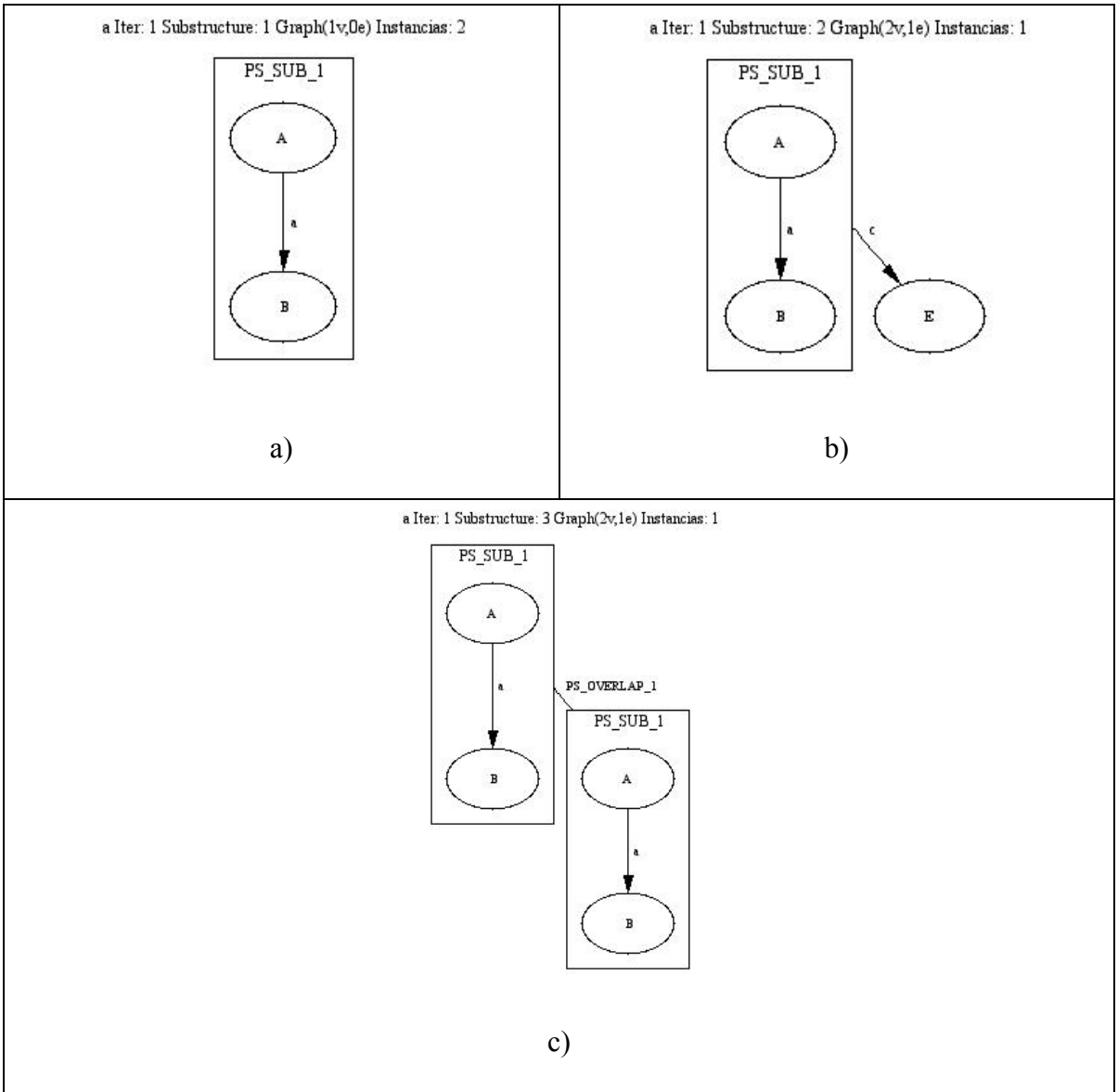


Figure 4.40. Limited overlap - discovered substructures.

4.4 Conclusion

In this chapter we have described the characteristics and functionality of our graph-based data mining tool, the Subdue system. We introduced a new algorithm named limited overlap. We presented some examples showing the functionally of the new algorithm using

an artificial dataset. Examples using data from the real world we will presented in chapter 6. These examples are developed by using two test contexts: a Puebla downtown population census from the year of 1777 and a Popocatépetl volcano database.

In the next chapter we introduce a prototype system implementing the proposed model for representing spatial data, non-spatial data and spatial relations among the spatial objects as a whole dataset using a graph-based representation.

Chapter 5

PROTOTYPE

A natural skill of people is the interpretation of visual data; therefore, this is a fact we must consider to get advantage during the data mining process. In [29] the authors say that a future direction for the *KDD* research field is the design and use of user interfaces: "one can create a query language which may be used by non-database specialists in their work. Such a query interface can be supported by a Graphical User Interface (*GUI*) which can make the process of query creation much easier".

The challenge consists in improving the capabilities for displaying the generated results (i.e. from a query or the data mining process) in a graphical mode. The idea is that if we are able to analyze the results in such a graphical way, we may give feedback to the user so that he can refine the analysis process and/or guide the direction for further study. This is the principle in relevance feedback (do an initial query, get feedback from the user, and then to incorporate information obtained from prior relevance judgments to redefine the query).

We developed a prototype system that provides a graphical user interface to perform the data mining phase (and data analysis) using our proposed graph-based representation for

spatial and non-spatial data. The analysis is implemented by using spatial and non-spatial queries and the data mining process is performed with the Subdue system, a graph-based data mining tool. In our research we used two test contexts to evaluate our proposal. The first one is a database storing data from the Popocatépetl volcano (see chapter 3.4 for details). The other one is a database containing data related to a population census from the year of 1777 in Puebla downtown as described in section 5.1.

5.1 Population Census from the year of 1777 in Puebla downtown

As our test context we have worked in the project "Habitar y vivir. Análisis del espacio habitacional de la ciudad de Puebla 1690-1890". This is a project directed by Dra. Rosalva Loreto López, a researcher in the Urban History domain. Our objective is to make use of data mining techniques for finding interesting relations and patterns between the population and habitation spaces in Puebla downtown by that time period. We argue that using our model we could find patterns involving non-spatial data (i.e. characteristics of people living in the zone), spatial data (i.e. distribution of the space), and relations between them (i.e. characteristics of houses based on people social status and/or number of people living in a house) in a single pattern.

Figure 5.1 shows the spatial structure we implemented for representing the spatial concepts in the census. The structure has 5 spatial aggregation levels. *Parish* (spatial concept for representing a physical area) is the upper spatial component. A *Parish* is defined by one or

more *Neighborhood*. A *Neighborhood* can involve several *Block*. A *Block* is defined by four *Street* and finally a *Street* gives the location of a *House*, where a family lives.

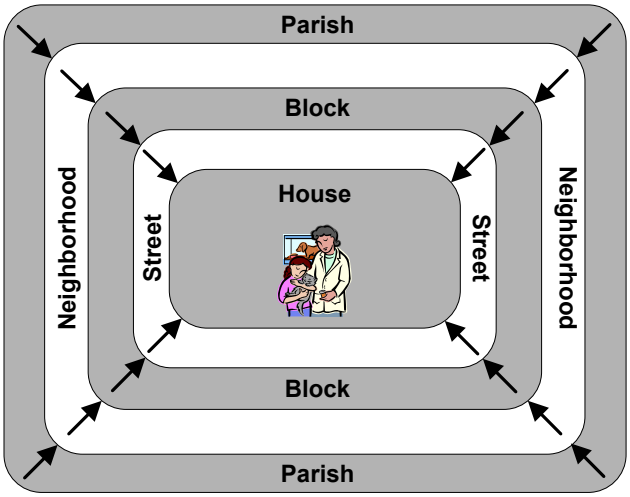


Figure 5.1. Representation of spatial concepts in the census from the year of 1777.

After defining the spatial concepts involved in our domain, we need a graph-based data representation model to describe our data as a graph. We must remember that this is a vital step since we work with a graph-based algorithm for the data mining step. Currently, we have implemented two graph-based data representations to model the non-spatial data.

These representations have three main components: (1) **HOUSE** involves the attributes describing a *House*. It contains one or more *Uh* (atomic physical area where a family lives). One or more families might inhabit in a *House*, but each family lives in an *Uh*. (2) **UH** involves the attributes describing the living space. (3) **MEMBER** involves the attributes describing a member of a family.

Figure 5.2 shows the first structure created for the population census. A description of this structure is as follows: a *Parish* contains one or more *Neighborhood*. A *Neighborhood* contains one or more *Block* or *Location* (spatial concept for identifying with precision a physical area -i.e. north of-). *Block* and *Location* contain one or more *Street*. A *Street* contains one or more **HOUSE**. A **HOUSE** has two attributes (*NCasa* -Id assigned to the *House*- and *NHabCasa* -number of *Uh* in a *House*-). A **HOUSE** contains one or more **UH**. An **UH** has several attributes describing it (*Uh* -Id assigned to the *Uh*-, *Etnicidad* -predominant ethnic group among the members of a family-, *TFamilia* -type of family, i.e. family with children-, *TUh* -type of *Uh*- and *NMiembrosFamilia* -number of members integrating a family-). An **UH** contains one or more **MEMBER**. Finally a **MEMBER** has several attributes describing it (*JFamilia* –family chief-, *TitPersona* -title of the member, i.e. don, doña-, *Nombre* -first name-, *Apellido* -last name-, *Sexo* -sex-, *EdoCivil* -marital status-, *Parentesco* -social relationship with respect to the family chief-, *GpoEtnico* -ethnic group- and *Edad* -age-).

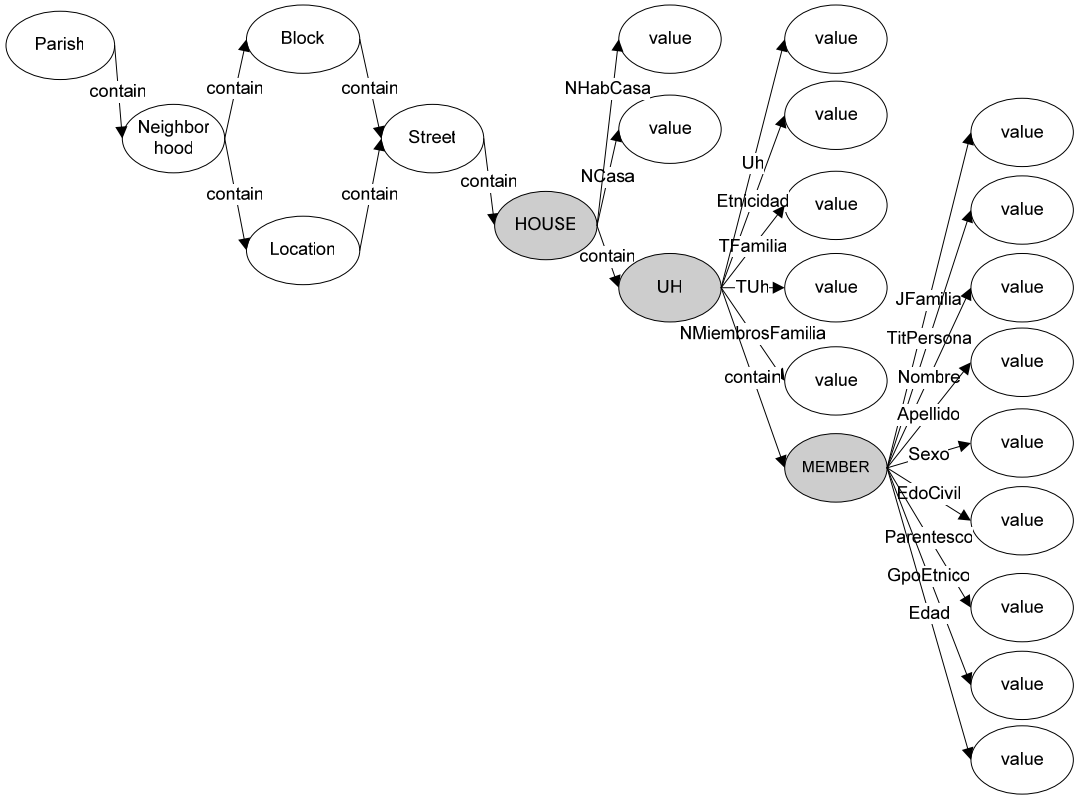


Figure 5.2. Model *A* to represent non-spatial data in the census from the year of 1777.

The second structure is presented in Figure 5.3. This structure can be read as follows: A **HOUSE** is the main piece in the structure. A **HOUSE** has several attributes describing it (*Parish*, *Neighborhood*, *Block*, *Location*, *Street*, *NCasa* and *NHabCasa*). A **HOUSE** contains one or more **UH**. An **UH** has several attributes describing it (*Uh*, *TUh*, *Etnicidad*, *TFamilia*, *NMiembrosFamilia*). In a **UH** lives (contains) one or more **MEMBER**, and finally, a **MEMBER** has several attributes describing it (*Jfamilia*, *TitPersona*, *Nombre*, *Apellido*, *Sexo*, *EdoCivil*, *Parentesco*, *GpoEtnico* and *Edad*).

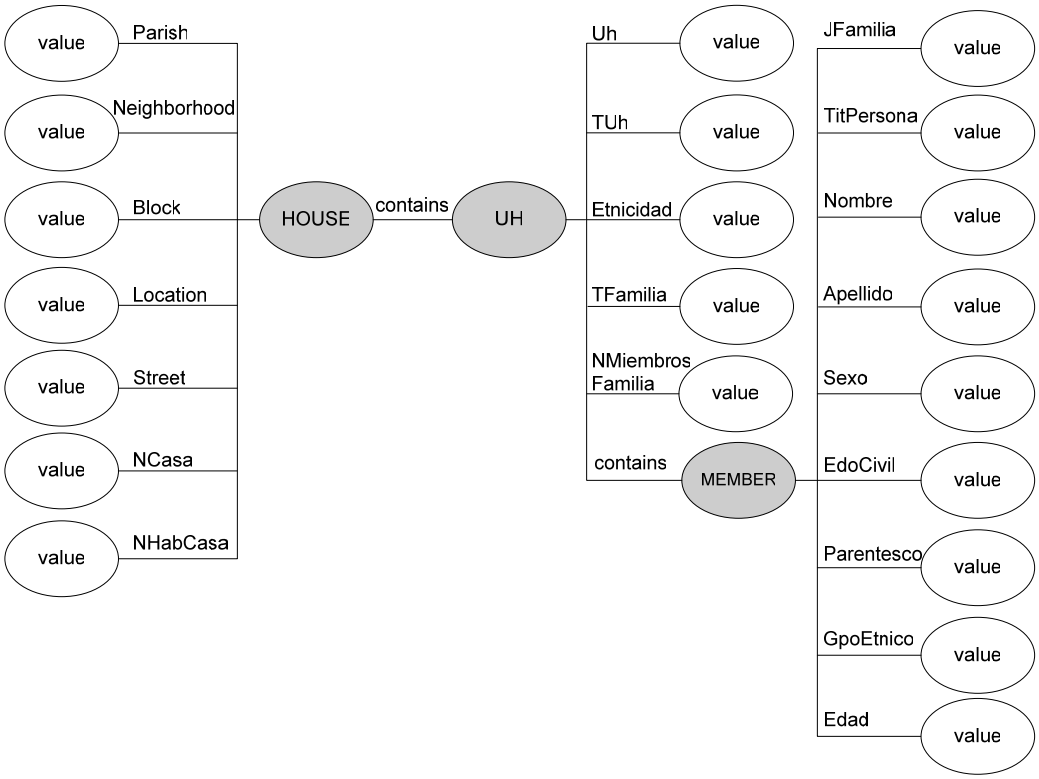


Figure 5.3. Model *B* to represent non-spatial data in the census from the year of 1777.

5.2 Modules

This section presents a prototype system developed for testing our graph-based data representation model for spatial data mining as proposed. The prototype is implemented in the Java programming language. We use the *Oracle DBMS version 9i* for storing and processing our data. Oracle has a module to manage spatial data named Oracle Spatial; we take advantage of the spatial operators, geometry functions and spatial aggregate functions implemented in the module for either identifying the objects in a region over a spatial layer or obtaining/validating the spatial relations between two spatial objects. The prototype is divided in seven modules described in the following subsections.

Data Preparation and Cleaning. The data preparation and cleaning phase is a step of the Knowledge Discovery in Databases (*KDD)* process. Our module helps the user to validate the data and creates the structures necessary to store it in the database. In this process, the data mining expert and the user must work together to identify the activities to be performed that are related to the process (i.e. to identify noise and remove it, treat missing values, etc). Once these activities have been defined, the process is transparent to the user when adding more data (belonging to the same database schema for the domain).

Query. We have implemented the interface shown in Figure 5.4 to allow the user creating non-spatial queries. The goal is to allow the user, making use of spatial and non-spatial attributes, to query the database and build graphs from the obtained results.

The *Query* panel allows the user for querying the database by using an *SQL-like* approach. The queries are created in real time and then submitted to the database for answering the user request. The results are presented to the user in two ways. The first uses a relational approach and the second is represented over a map.

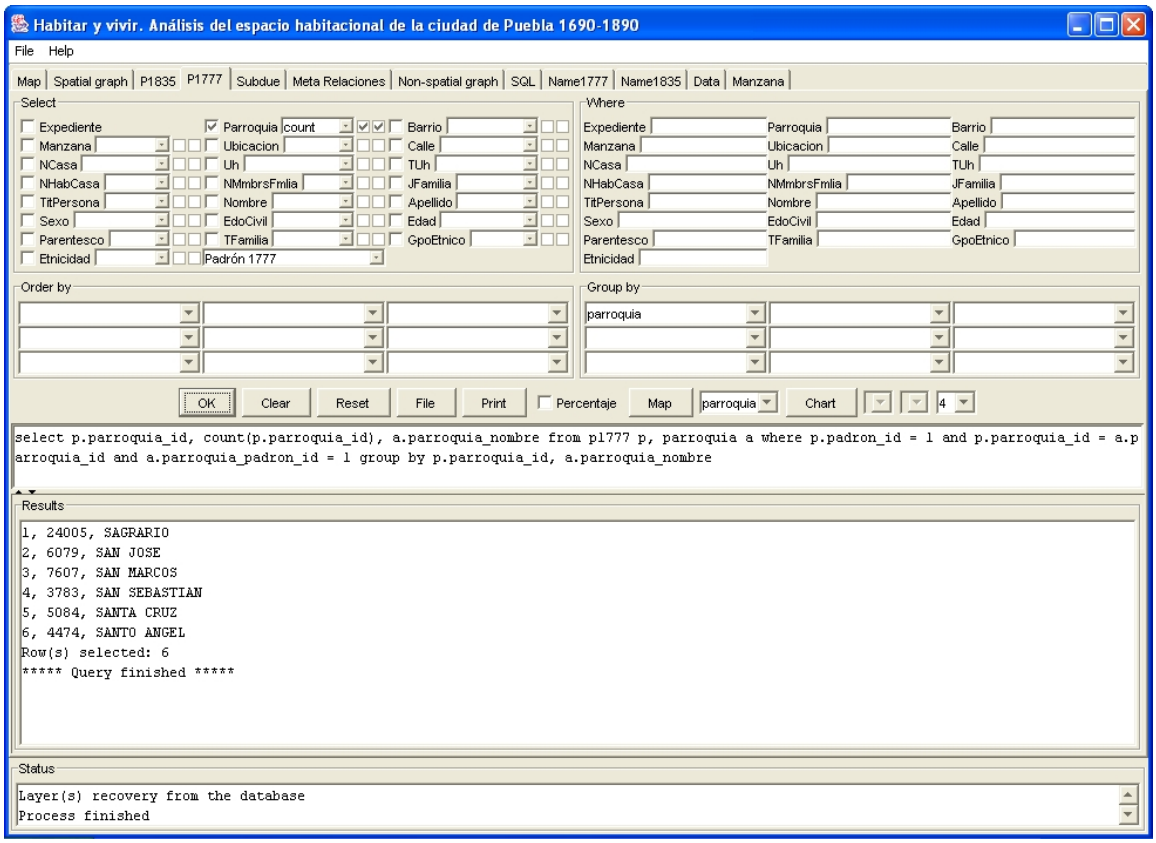


Figure 5.4. The query panel.

The interface is divided in 2 sections, the *Control* and the *Results* sections. The *Control* section is divided in 4 subsections: *Select*, *Where*, *OrderBy* and *GroupBy*. The Results section is divided in 2 subsections: *Query visualization* and *Generated results*.

In the *Control* section the user creates the query. The query is created by specifying the 5 most usual clauses in a *SQL* statement: *Select–From-Where-OrderBy-GroupBy*. The first two elements are mandatory, while the last three are optional.

The *Select* subsection presents the twenty-two fields that compose the core of the database. Twenty one of these fields have four options implementing query functionalities as follows: *option1 – attribute name – option2 – options3 – option4*. The first option is used to indicate if the field will be included in the query. The second option contains the keywords used to implement, currently, five grouping functions: "*count*", "*count(distinct())*", "*distinct*", "*max*" and "*min*". These keywords are used to group the data by the field(s) selected in the *GroupBy* subsection. Options three and four are used to indicate if the descriptive attributes associated to the field will be included in the query. If the first option is not selected then all the other options are ignored. Additionally, the Select subsection has an item containing the name of the schema (i.e. 1777 census) to create the "*from*" clause of the query.

The *Where* subsection is used to indicate the set of conditions, by field, for restricting the rows to be selected (the dataset returned by the query). A condition specifies a combination of one or more expressions composed of attribute names, attribute values, and logical operators. These conditions are entered manually by the user, so knowledge about the domain is required.

The *OrderBy* subsection allows the user to indicate if he/she wants to sort the rows returned by the query and which field(s) will be used to perform the operation. We can sort the rows returned using the twenty-two fields composing the database and we can also implement combinations of these elements (i.e. sorting by the *Name* and *Sex* fields).

The *GroupBy* subsection was implemented to allow the user to group the selected rows based on the value(s) of each row and return a single row with a summary of the

information for each group. We can group the rows using twenty-one of the twenty-two fields of the database and we can also implement combinations of these elements (i.e. grouping by the *Parish* and *Sex* fields). The grouping aggregation function (i.e. *count* or *sum*) is chosen in the *Select* subsection. As we already mentioned, we have implemented five grouping functions.

Our prototype system creates queries by selecting the different fields and options contained in the four previous sections. This query is created in real time, displayed in the *Query visualization* area and then submitted to the *DBMS* for processing. The obtained result is displayed in the *Generated results* area; it is presented by using a relational approach (rows and columns).

SQL. Using this interface the user has the freedom to create a query by typing it directly (in manual form). The principle is the same as we described in the *Query* panel, we want to create queries and with the results of the queries we create graphs (based in our model). These graphs will be the data source for our data mining algorithm so that we can find patterns that allow us to understand and describe our data.

In the *Query* panel the user creates the query by using a graphical interface but if he/she wants to modify it, it is not possible. Each time the user creates a query in the *Query* panel, the created query is copied to the *Query visualization* area in the *SQL* panel so if the user wants to enhance or modify the query he/she is able to do it. The generated results are presented to the user in the *Generated results* area. Both interfaces (*Query* and *SQL* panels) provide the user with the capability to send the generated results to a text file or a printer.

Map. Other way to create graphs in the prototype is using the *Map* panel. In this case the interface displays in a graphical way the spatial layers stored in the database (see Figure 5.5).

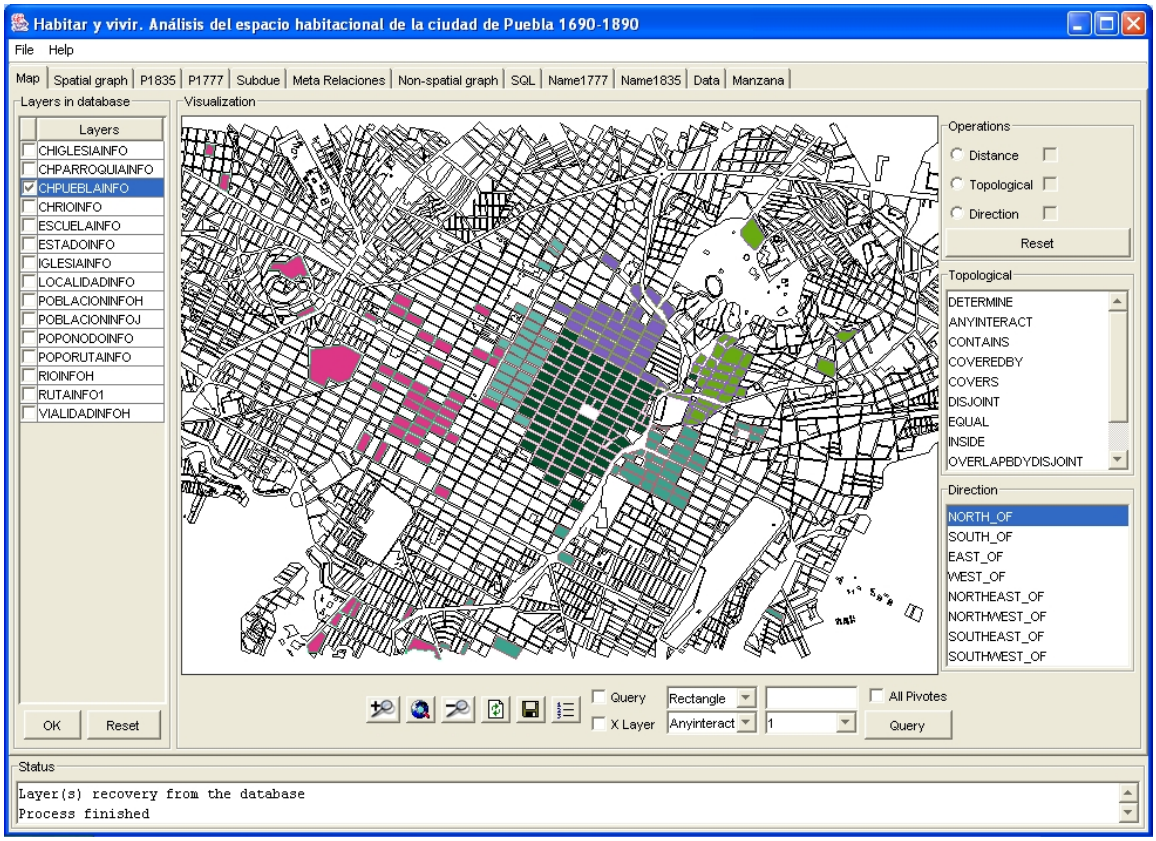


Figure 5.5. The map panel.

By using the interface, the user can delimit the set of spatial and non-spatial data that will be used for creating the graphs. Some times the user wants to analyze only some regions in a spatial layer so it is not necessary to include all the data in the graph. Additionally, we have to take into account that if we have a huge database we will build huge graphs and this feature has a direct impact over the data mining algorithm, so this is an important issue we

have to face. A solution for facing the problem of creating huge graphs is delimiting the set of elements to be included in it by using *selection windows* as we mentioned in chapter 3.

A second method for delimiting the set of elements to be considered while creating a graph is by using the results of the non-spatial queries created and processed in the *Query* and *SQL* panels. This is implemented by a process that identifies the spatial objects involved in the results generated by a non-spatial query (i.e. a query computing the number of people, grouped by *Sex*, living in each *Parish* in the census from the year of 1777 so we can select and show the *Blocks* or the *Streets* belonging to each *Parish* over the map).

The interface is divided in four sections: *Visualization area*, *Layers in database*, *Operations control* and *Map control*. The *Layers in database* section displays the name of the spatial layers stored in the database. The component is also used for selecting and identifying the spatial layers to work with.

We also include information about the spatial relations to be considered in the graph. The *Operations control* section includes the operations (*Topological*, *Distance* and *Direction*) implemented to validate the spatial relations among spatial objects. In the case of the topological relations we have implemented the validations supported by the Oracle Spatial module.

The *Map control* section has five buttons implementing the *Zoom in*, *Zoom out*, *Show all*, *Reset all* and *Save map* operations. The *Visualization area* works in two operational modes: *Query mode* for creating *selection windows* and *Zoom mode* for implementing the *Zoom in*

and *Zoom out* operations. The option "*X Layer*" is used to indicate if the validation of spatial relations among spatial objects will just be among objects belonging to different spatial layers (i.e. objects belonging to the *Parish* and *Neighborhood* layers) or among all objects belonging to all layers. The last two options are used to control the characteristics of the *Selection window*: a *Selection window* tool can be a *Rectangle* or a *Circle*, and we can specify if we want to select just the elements inside the *Selection window* or the elements inside and touching its border.

Once the user has selected the working area, the following steps are identifying the objects inside the area and validating the spatial relation(s) among them. We have implemented the validation of topological, distance and direction relations; only the objects meeting the relation(s) chosen by the user will be candidates to become objects in the graph.

Spatial Graph. The next task is to create the graph. First, the user must select the non-spatial attribute(s) describing the spatial objects in the graph (see Figure 5.6). Remember that the spatial objects and spatial relations among the objects that will be included in the graph were selected in the *Map* panel.

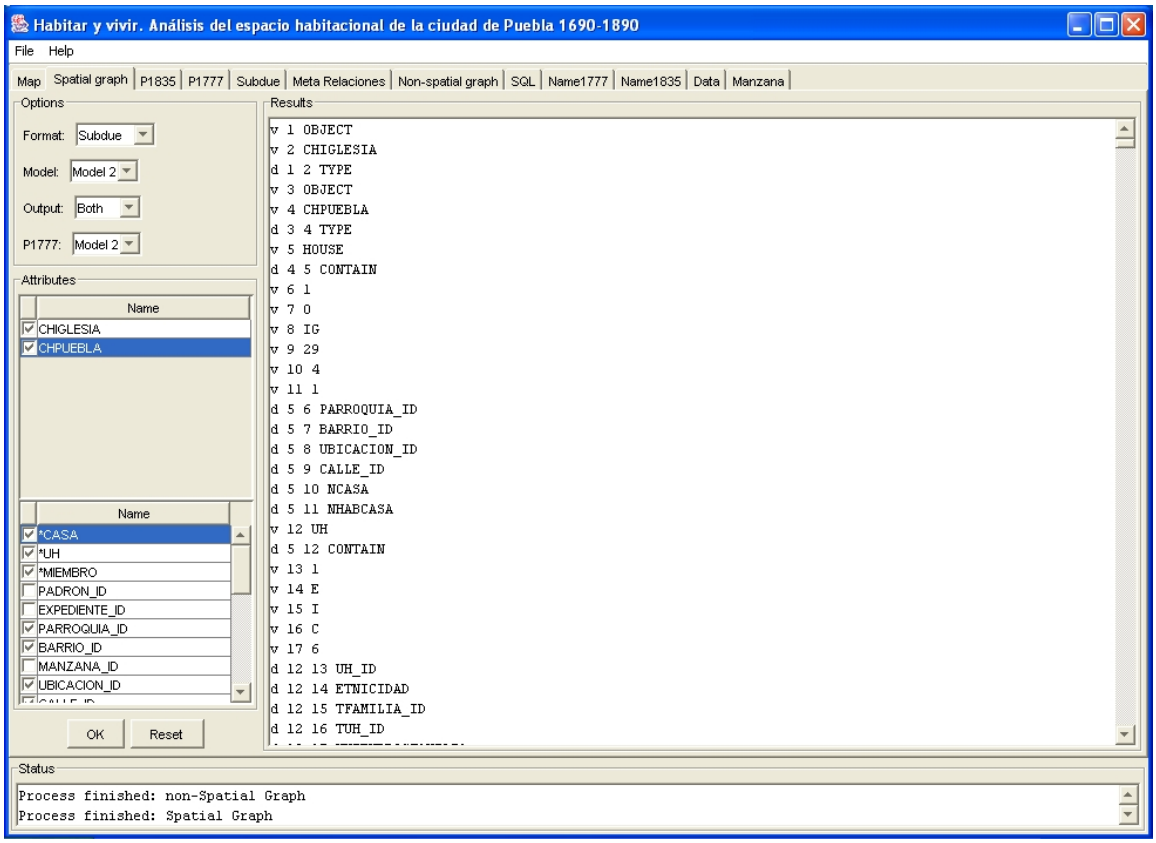


Figure 5.6. The spatial graph panel.

By using the interface, the user can select the non-spatial attributes of each spatial layer that he wants to work with. For example, if the user works with the spatial layers *A* (with five attributes) and *B* (with three attributes), he/she can select attributes one and two from layer *A* and attributes two and three from layer *B*. Currently, the user has to select the attribute(s) that will be related to the spatial objects, but as we have mentioned, we are working to enhance this functionality of the *Query* and *SQL* panels so that we can create non-spatial queries and then use the results for selecting spatial objects and then creating a graph.

From the *Graph characteristics* section the user defines the format, model and output device characteristics for the graph. The graph generated by the system can be created following the Subdue or the Graphviz [19] layouts. In the first case the graph is created for feeding the Subdue system, and in the second case it is created for visualization purposes. Currently, we have implemented five graph-based representation models in our prototype. Each model expresses a representation proposal for creating graphs involving the three basic elements found in a spatial database (spatial, non-spatial data, and spatial relations among spatial objects). The resulting graph can be visualized on the screen or stored in a text file. The last option was implemented in order to provide the Subdue and Graphviz systems with their corresponding input files.

We can see at the right side in the Figure 5.6 an example of a created graph using the Subdue layout. This sample graph was generated from 2 spatial layers (i.e. chiglesia and chpuebla). From each of them the user selected one or more attributes that were related to the spatial objects. Figure 5.7 presents a fragment of the same graph but this time it is drawn by using the Graphviz system.

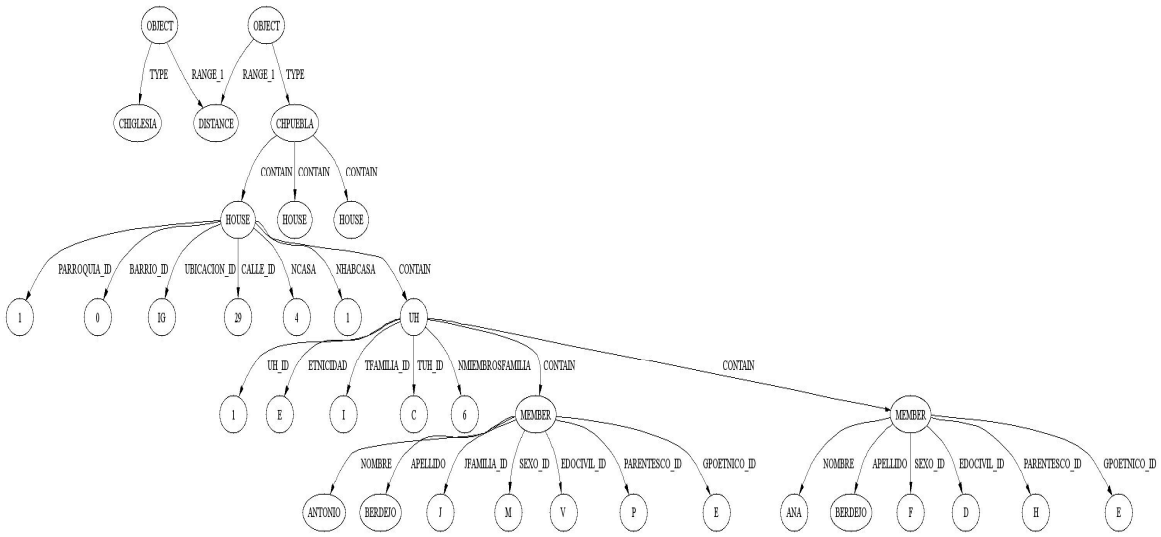


Figure 5.7. Graph representation of processed data.

Non-Spatial Graph. Focusing in the "Habitar y vivir. Análisis del espacio habitacional de la ciudad de Puebla 1690-1890" project we have implemented an interface for allowing the user the creation of graphs based in the two structures developed for representing the population census from the year of 1777. These graphs are created using only non-spatial data. Our objective for implementing graphs with this characteristic is to have metrics that allow us to compare and to evaluate the results generated by the data mining algorithm when we use graphs containing spatial data, non-spatial data, and spatial relations at the same time against graphs containing only non-spatial data.

The interface implemented is shown in Figure 5.8. The user selects the non-spatial attributes and defines the settings that will be used for creating a query (as in the spatial graph). The result obtained from the query is used for creating the non-spatial graph.

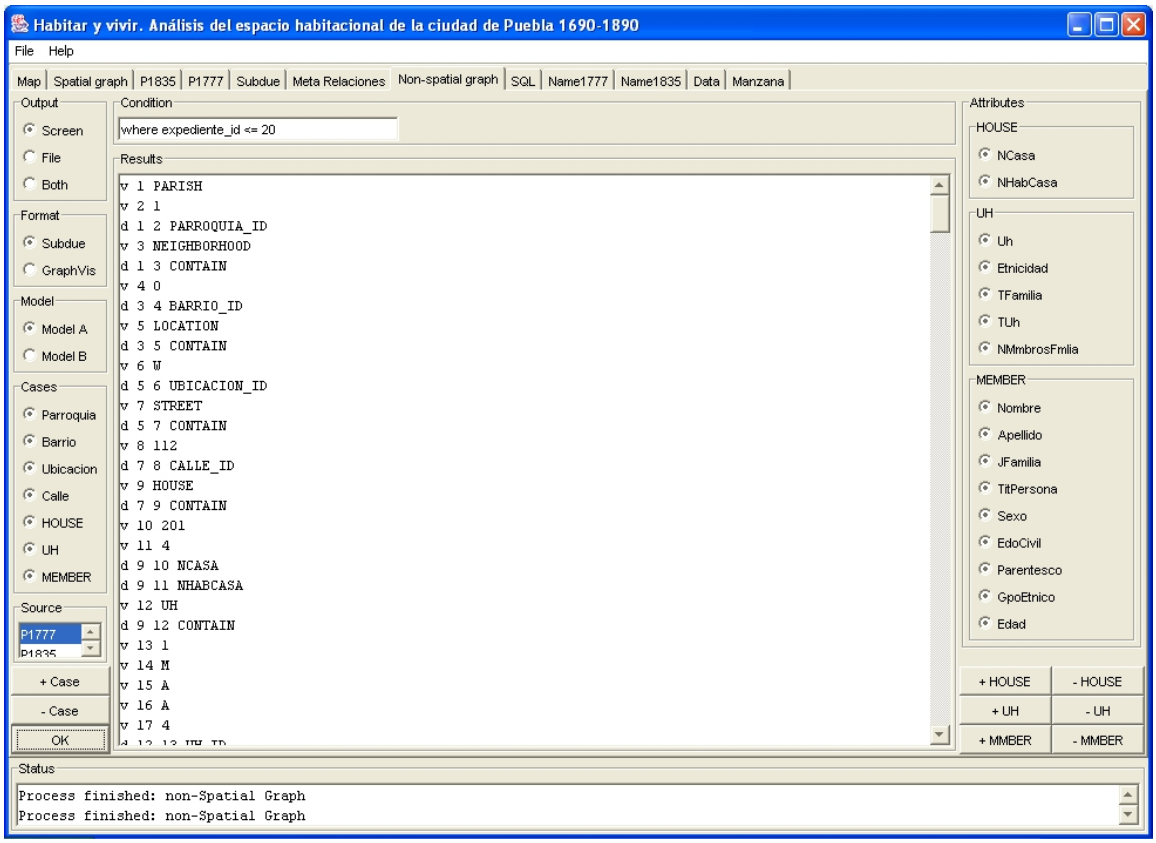


Figure 5.8. The non-spatial graph panel.

Subdue. The *Subdue* panel contains the interface developed for calling the Subdue system (see Figure 5.9). In order to run Subdue, the user must select the corresponding text file (a file containing a graph) and define the parameters that will guide the Subdue's substructure discovery system for finding substructures.

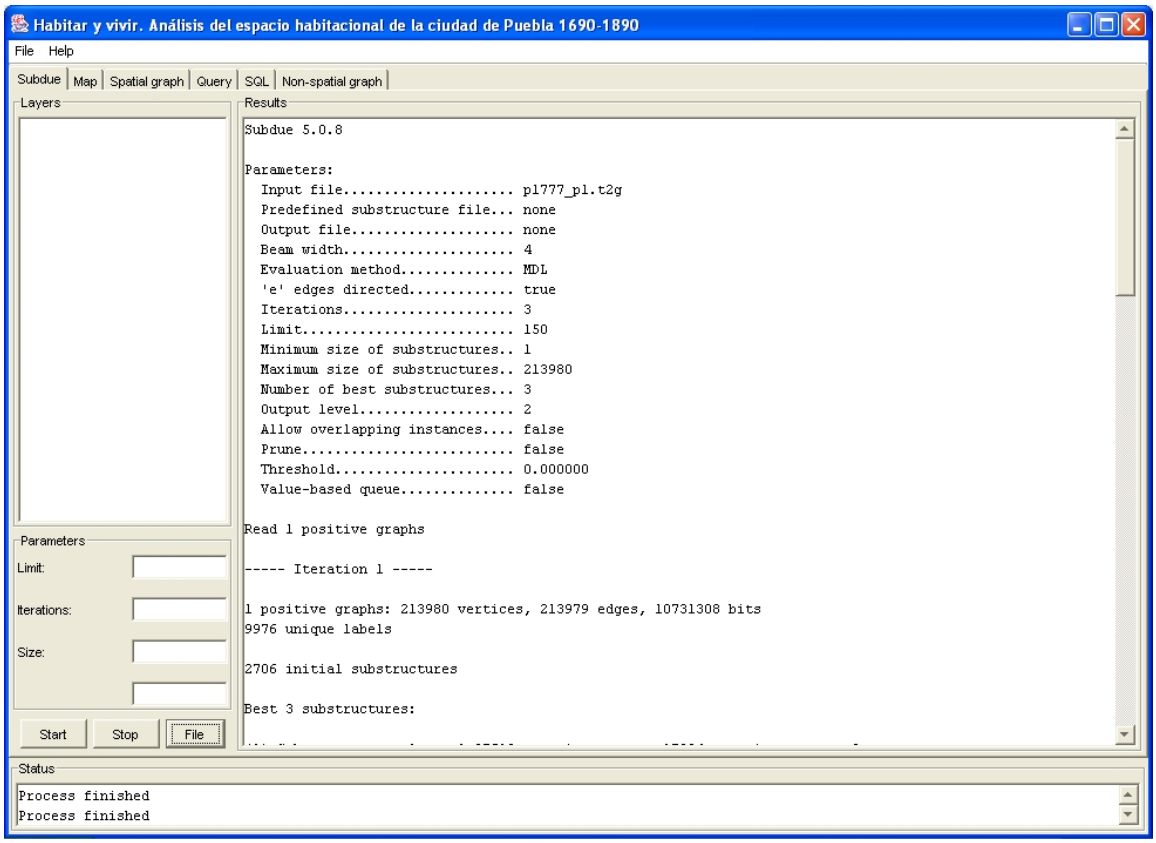


Figure 5.9. The Subdue panel.

Figure 5.10 shows an example of a Subdue's standard output (only for one substructure) for displaying the discovered substructures (i.e. patterns) from the input data. Since our definition of instances and substructures (see chapter 4.1 for details) the Subdue's output is also a graph, in our example, the graph can be read as follows:

- Substructure value = 1.01706. Represents the MDL value of the substructure.
- Pos instances = 3865. This value tells us how many instances of the substructure exist in the input graph.
- Graph (2*v*, 1*e*). Number of vertices ("v") and edges (in our example directed edges "d") compose the graph.

- "v 1 SUB_4". First vertex labeled as SUB_4 of the graph.
- "v 2 X". Second vertex labeled as X of the graph.
- "d 1 2 ETNICIDAD". Directed edge from vertex 1 to vertex 2 labeled as ETNICIDAD.

```
Substructure: value = 1.01706,
pos instances = 3865,
neg instances = 0
Graph(2v,1e):
v 1 SUB_4
v 2 X
d 1 2 ETNICIDAD
```

Figure 5.10. Example of Subdue's standard output.

As we have already commented, Subdue is a system that finds substructures in a hierarchical way, that is, a substructure found in a previous iteration can appear in a new iteration. When this happens, those substructures are represented by a vertex labeled as *SUB_x*, which represents the best substructure discovered at iteration "*x*".

In our example (Figure 5.10), *SUB_4* is itself a substructure defined by 2 vertices and 1 edge where its first vertex is labeled as *SUB_2*. This means that the definition of *SUB_4* is composed by the definition of *SUB_2*. Substructure *SUB_2* is defined by 2 vertices and 1 edge where its first vertex is labeled as *SUB_1*. Again, this means that the definition of *SUB_2* is composed by the definition of the previously discovered substructure *SUB_1*. Finally, *SUB_1* is a substructure defined by 2 vertices and 1 edge.

As we can see, the lecture and interpretation of the substructures discovered by Subdue may be a complicated task. We have implemented a parsing function for reading a text file

containing the results generated by Subdue, so we can create the necessary data structures for presenting to the user the same results but using an easier way to read it as we show in Figure 5.11 (the figure presents the example described in Figure 5.10). This layout is created by using the Graphviz system and is saved as a JPEG image file but we can save it in any output format supported by Graphviz. The goal is to improve the way the user can read and interpret the results generated by Subdue.

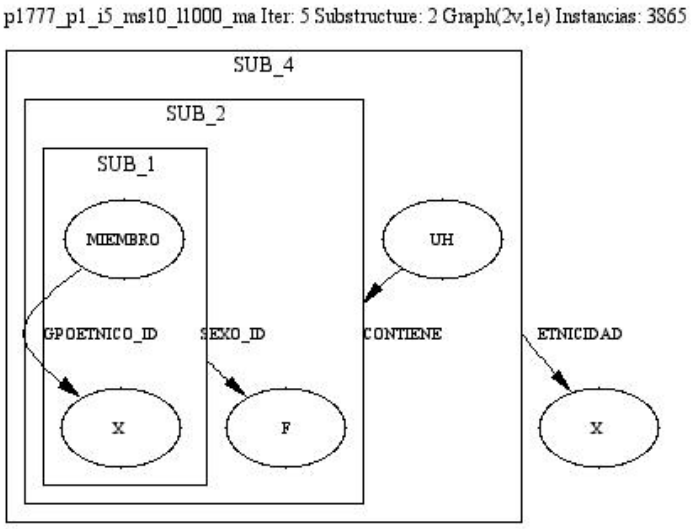


Figure 5.11. Layout for reading the Subdue's discovered substructures.

5.3 Conclusions

As test context we have designed and built a spatial database to store both a population census from the year of 1777 in Puebla downtown and a map representing the blocks in the zone. This data are part of a project directed by Dra. Rosalva Loreto López, a researcher in the urban history domain.

We have developed a prototype system implementing our model to represent together spatial data, non-spatial data and the spatial relations among the spatial objects. The prototype allows the user to select the spatial layers to work with, to create spatial and non-spatial queries that will be used to select the spatial objects that will be included in the graph. For each spatial layer the user work with, he/she has the capability to select the attributes that will be related to the spatial objects in the graph.

We have also implemented a visualization tool which helps us to display in a graphical way (by using the Graphviz system) the hierarchical discovered substructures by Subdue.

In the next chapter we present four use-cases showing the applicability of our methodology for modeling and mining spatial data mining using a graph-based representation.

Chapter 6

RESULTS

This chapter presents four use-cases of our methodology for modeling and mining spatial data using the proposed graph-based representations. For this purpose, we used two spatial databases as our test contexts. The first database contains data related to a population census from the year of 1777 in Puebla downtown and the second one is a database storing data from the Popocatépetl volcano.

The use-cases described were implemented based on the following premises: evaluating the graph-based proposal for modeling and mining spatial data, evaluating the limited overlap feature, and evaluating the discovered knowledge with the support of a domain expert. Therefore, the three use-cases presented in this chapter were implemented based on the following methodology:

1. Selection of the spatial layers to work with.
2. Selection of the spatial relations that will be validated among the spatial objects.
3. Selection of the non-spatial attributes that will be related to the spatial objects in the graph.

4. Mining the graph using the no overlap, standard overlap and limited overlap features.
5. Evaluation of discovered patterns.

6.1 Population census from the year of 1777 in Puebla downtown

As we have already mentioned, our first test domain is a spatial database containing data of a population census from the year of 1777 (see chapter 5 for details). Figure 6.1 shows a fragment of the "*chpuebla*", "*chiglesia*" and "*chrio*" spatial layers used in the use-cases.

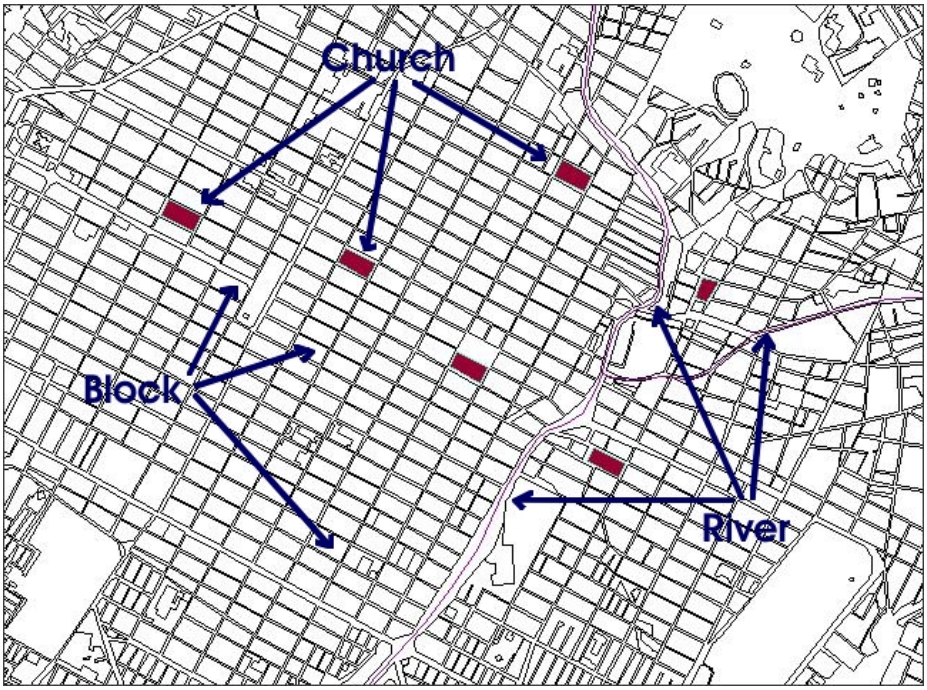


Figure 6.1. Population census from the year of 1777 in Puebla downtown.

The *chpuebla* spatial layer (shown in white color) represents blocks in Puebla downtown; this layer is related to a population census from the year of 1777 as non-spatial data. The

chiglesia layer (shown in red color) contains representative churches for each parish in the zone. The *chrio* layer (shown in green color) represents a river crossing Puebla downtown.

It is important to remark that a parish is a spatial object grouping several blocks in the zone. Each parish has a church as its agglomerative element (people used to live around a church). Figure 6.2 shows the 6th parishes (each shown in a different color) and their representative church:

1. El Sagrario.
2. San José.
3. San Marcos.
4. San Sebastián.
5. Santa Cruz.
6. Santo Angel.

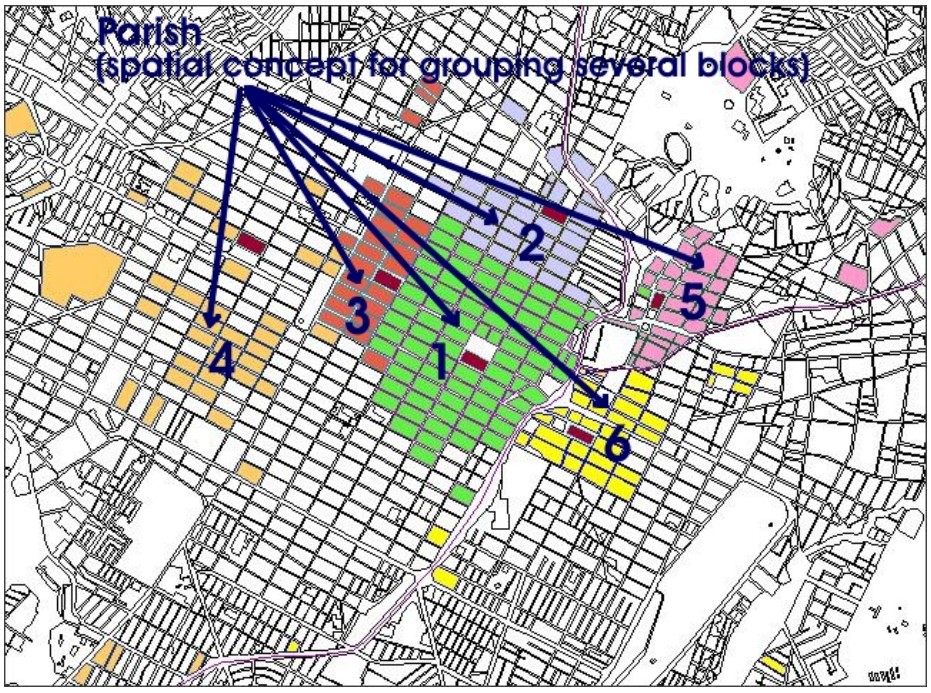


Figure 6.2. Parishes in Puebla downtown from the 1777 year.

All use-cases presented in this section were developed using all graph-based models (currently five models). However, the description of the generated results corresponds only to model #2 (named single replication of relations types, complete information). In section 6.2 we present a comparison among the generated results by each proposed model using a Popocatépetl volcano database. Some of the discovered patterns were used by the domain expert to validate facts already known (i.e. distribution of the population in the census) and other allows him to know unknown relationships among spatial objects and non-spatial attributes in the census (i.e. common characteristics of people living along the two borders of the river crossing Puebla downtown).

6.1.1 Use-case: El Sagrario

Suppose we want to know what people have in common in the spaces within a radius of 150 meters from the representative church in parish #1 (El Sagrario). Our experiment will be focused to find regularities related to the following issues:

- Number of habitation spaces in a house.
- Members of a family.
- Type of family.
- Ethnic group of each family member.

The guideline to select a radius of 150 meters from the church is that this value allows us to include in our sample dataset at least one block in all directions around the church as we show in Figure 6.3. Thus, by using the prototype system we selected the *chpuebla* and

chiglesia spatial layers. Once we selected the spatial layers to work with, the following steps are to select the spatial relation(s) to be validated among the spatial objects and the non-spatial attributes that will be related to the spatial objects in the graph. The selected parameters were the following:

- Spatial layers: *chpuebla* and *chiglesia*.
- Pivot: representative church in parish #1.
- Spatial relation: distance.
    - Value: 150 meters within a radius from the representative church to the blocks.
- Spatial graph-based model: model #2.

The generated graph was composed of 24,167 vertices, and 24,166 edges according to the proposed graph-based model structure.

Figure 6.3. Blocks 150m. from representative church, parish "El Sagrario".

This graph was used as input data to the Subdue system. For the experiment we used the following Subdue's parameters:

- Predefined substructure: yes (we used "*UH CONTAIN MEMBER*" since these elements are grouping components in our graph-based model for representing non-spatial data in the census). See chapter 5.1 for details.
- Overlap: yes.
- Limited overlap: no/yes (first, we used standard overlap; next, we used limited overlap).

Once Subdue completed the mining process, the generated results (patterns) were evaluated by the domain expert. Figure 6.4 shows three examples the discovered patterns using the standard overlap option. According to the lecture of the results, the expert's opinions were based on the following issues: the patterns are based on the population distribution schema in the census from the year of 1777 (large population inhabits in parish #1). 65% of the population, in this area, did not given its ethnic group, this can be interpreted in demography history domain, as a possible dissolution of the racial element for grouping people (creation of groups or classes) in benefits of alternative grouping parameters such as salary, family networks (how they lived and whom they lived with), consumption levels, type of house. 16% said to be "Spanish". People lived based on the model "Jefe con Familiares y Agregados".

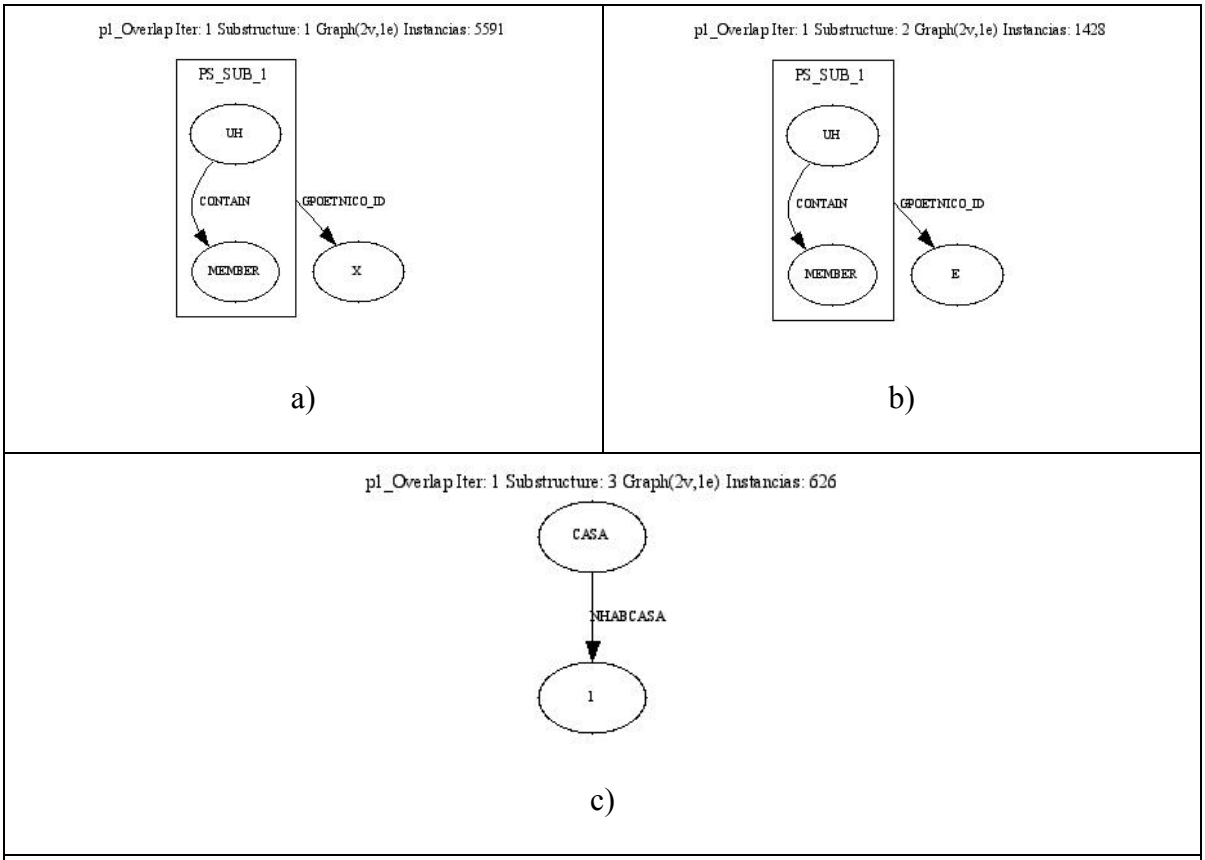


Figure 6.4. Examples of discovered patterns by using standard overlap in use-case "El Sagrario" (1).

The next step in our experiment was to evaluate our limited overlap proposal in Subdue. Thus, for the limited overlap, we proposed to allow overlap only in vertices representing the ethnic group of the family members. These vertices are used to guide the pattern-oriented search.

The discovered patterns by using the standard and limited overlap features, for this example, were slightly different. Figure 6.5 presents three examples of discovered patterns using the limited overlap. For example, both cases reported as the first substructure a pattern telling us "Undefined" is the predominant ethnic group in the dataset. The second discovered substructure, for both cases, tells us that "Spanish" is the next predominant

ethnic group. In the case of the third substructure, using standard overlap Subdue reported a pattern related to the number of habitation spaces in a house, but through limited overlap Subdue found a relationship among the "Undefined" ethnic group and the family type "Jefe con Familiares y Agregados". An interpretation of these results is that limited overlap reports in all discovered substructures a vertex representing ethnic group because we oriented the search over this type of vertices when we specified that vertices representing ethnic group were allowed for overlap.

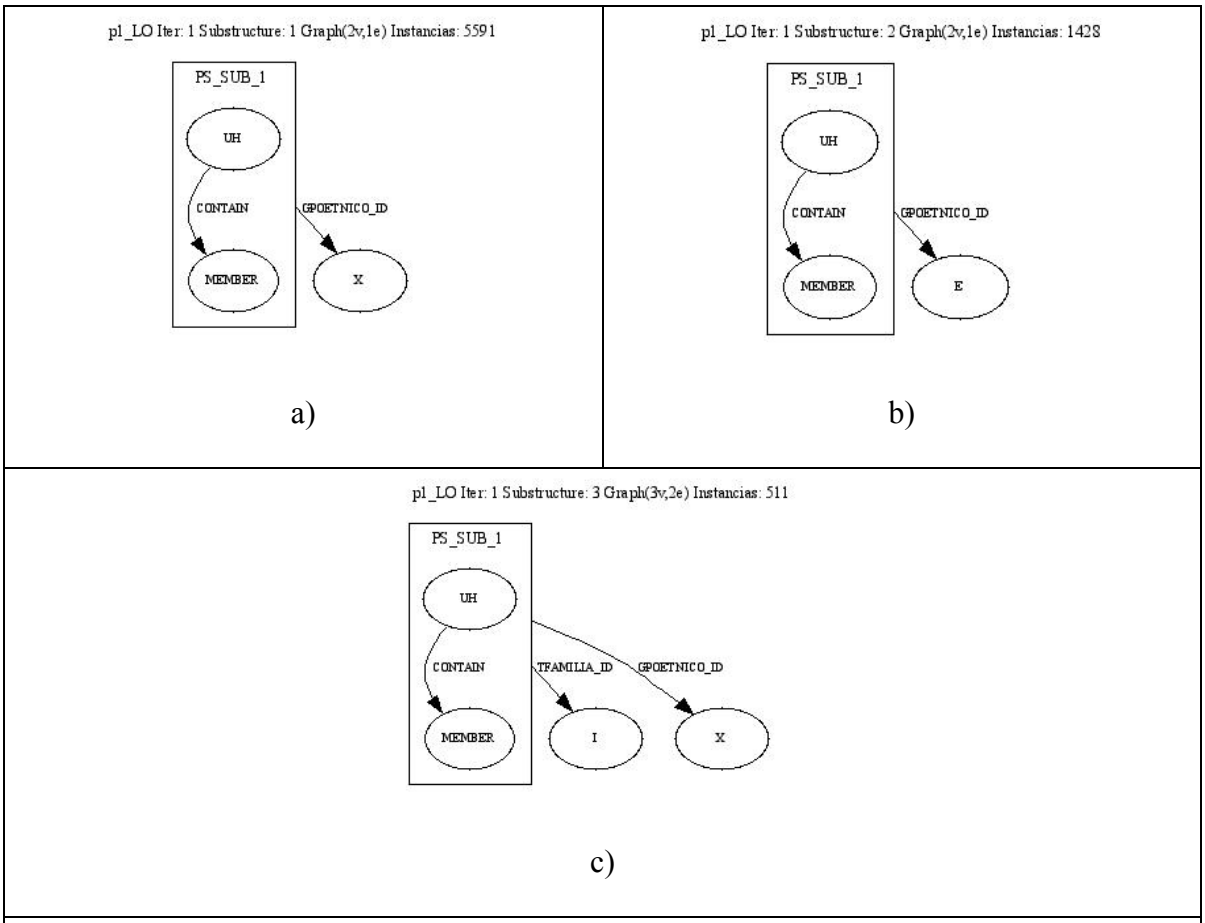


Figure 6.5. Examples of discovered patterns by using limited overlap in use-case "El Sagrario" (1).

Since our intention is to prove our objective of a processing time reduction by using the limited overlap feature, Figure 6.6 presents the processing time comparison chart for the

experiment. In the figure, time by using standard overlap is shown in pink color (193,426 seconds). On the other hand, processing time by using limited overlap is shown in yellow color (36,042 seconds). As we can see, we obtained a time reduction gain about 81.37% using limited overlap.

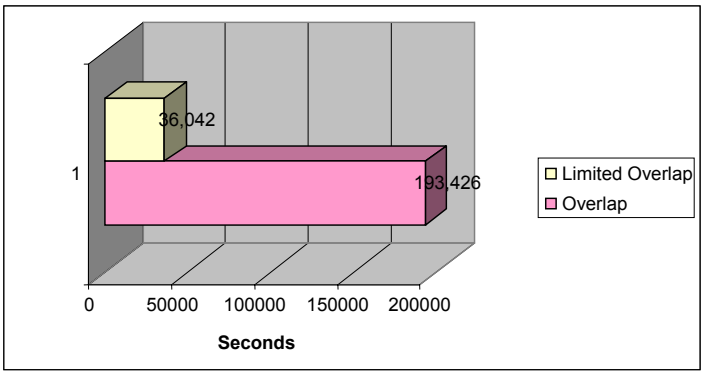


Figure 6.6. Processing time standard vs. limited overlap: use-case "El Sagrario" (1).

Now, we are going to modify slightly our study zone to show the way we can use two or more spatial relations in the experiments. For instance, to use two spatial relations belonging to the same type (i.e. topological), or to different types (i.e. topological and distance). Suppose we want to search for regularities about people and habitation spaces in a radius of 150 meters from the same representative church in parish #1, but this time, we only want to evaluate blocks located on the North side as shown in Figure 6.7. Our experiment will be focused over the same discovery issues as the previous test.

By using the prototype system we selected the following parameters:

- Spatial layers: *chpuebla* and *chiglesia*.
- Pivot: representative church in parish #1.

- Spatial relation: distance.
    - Value: 150 meters.
- Spatial relation: direction
    - Value: North.
- Spatial graph-based model: model #2.

The generated graph was composed of 12,021 vertices, and 12,026 edges according to the proposed graph-based model structure.

Figure 6.7. Blocks 150m. North side from representative church, parish "El Sagrario".

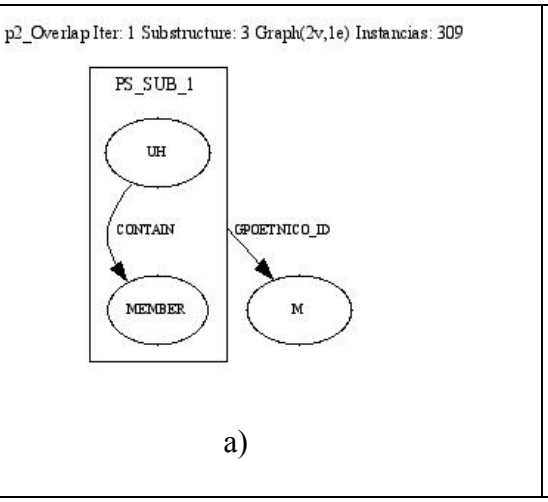
Such as the previous example, the graph was used as input to Subdue. For the experiment we selected the following Subdue's parameters:

- Predefined substructure: yes (we used "*UH CONTAIN MEMBER*" since these elements are grouping components in our graph-based model for representing non-spatial data in the census). See chapter 5.1 for details.
- Overlap: yes.
- Limited overlap: no/yes (first, we used standard overlap; next, we used limited overlap).

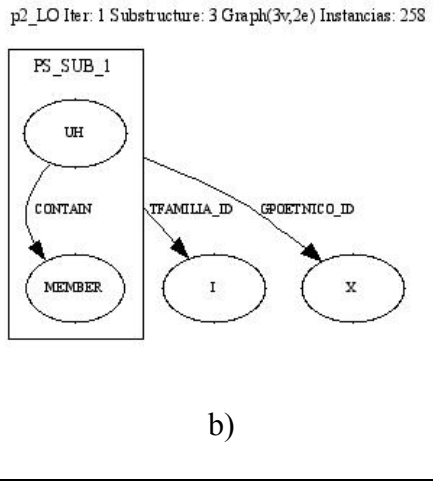
Once Subdue completed the mining phase, the generated results were evaluated by the domain expert. The expert found the following facts: the predominant ethnic group in the area remains as "Undefined" because of the proximity to the parish center and the continuous racial and social interchange from the North side. As consequence, they share the same family type structure. The "Spanish" population is the most important (15%) and the next one is "Mestizos" (7.13%). The family nucleus which includes "other people living with" (added people) employs "Mestizos" as subordinated workers (i.e. waiters and salesmen). It is important to remark that they do not employ "Indígenas" (maybe because they do not speak the Spanish language and their limited cultural level).

This is the domain expert evaluation but we also needed to evaluate how the new limited overlap feature worked. Therefore, the same experiment was performed using the standard and then the limited overlap. For limited overlap, the vertex allowed for overlap was the same as the previous test. The discovered patterns by using standard and limited overlap were very similar to those obtained in the first test. In fact, the first and second reported substructures were the same although the third one was different (see Figure 6.8). By using

standard overlap Subdue reports “Mestizos” as the third predominant ethnic group. Limited overlap reports a relationship among the “Undefined” ethnic group and the family type “Jefe con Familiares y Agregados”. This last pattern was the same one as reported in the previous test.



a)



b)

Figure 6.8. Examples of discovered patterns in the use-case “El Sagrario” (2)

Figure 6.9 shows the processing time comparison chart by using the standard and limited overlap features. In this figure, the time taken when using the standard overlap feature is shown in pink color (30,532 seconds) while the processing time of using limited overlap is shown in yellow color (2,471). It is important to note that we obtained a time reduction gain about 92% in our experiment.

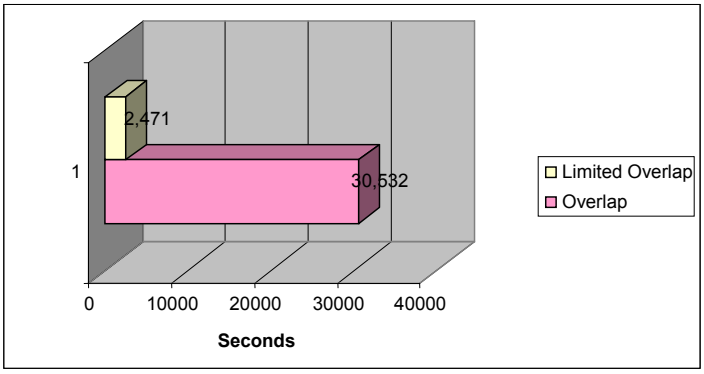


Figure 6.9. Processing time standard vs. limited overlap: use-case "El Sagrario" (2).

6.1.2 Use-case: People living along the borders of the river crossing Puebla downtown

As we mentioned at the beginning of this chapter, a river crosses Puebla downtown. Suppose that we are interested about knowing characteristics about family types and ethnic groups of people living along the borders of the river.

For this, we defined as our study area all blocks located at most 50 meters from the borders of the river as shown in Figure 6.10. We selected this distance since it allows us to select, both side, at least one block along the entire border of the river. We used the following parameters in our use-case:

- Spatial layer: *chpuebla* and *chrio*.
- Pivot: the river.
- Spatial relation: distance.
    - Value: 50 meters.

- Spatial graph-based model: 2.

The generated graph was composed of 16,597 vertices, and 16,596 edges according to the proposed graph-based model structure.

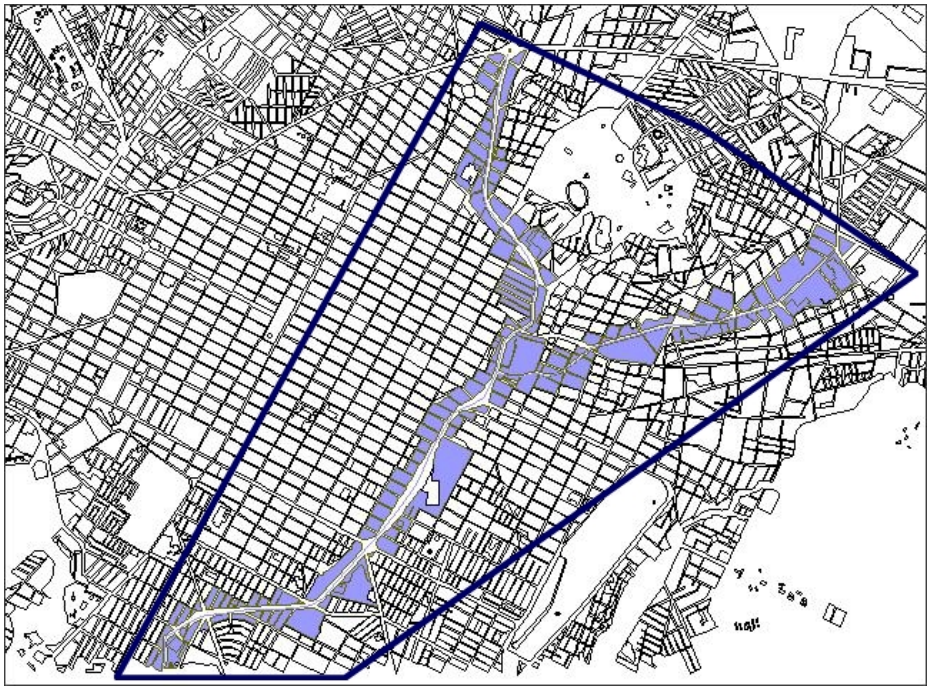


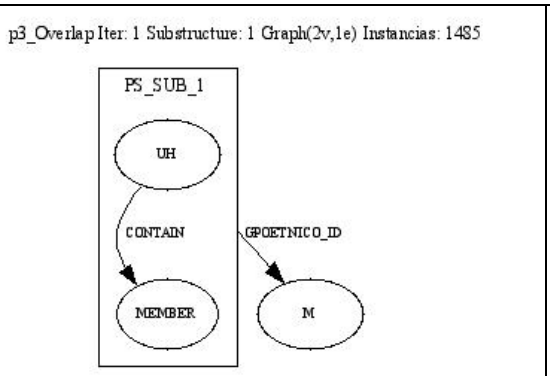
Figure 6.10. Blocks 50m. from river crossing Puebla downtown.

The created graph was used to feed the Subdue system. For this test we selected the following Subdue's parameters:

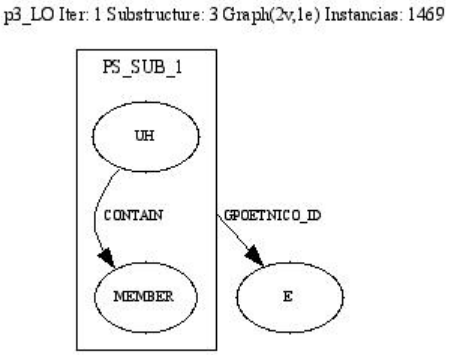
- Predefined substructure: yes (we used "*UH CONTAIN MEMBER*" since these elements are grouping components in our graph-based model for representing non-spatial data in the census). See chapter 5.1 for details.
- Overlap: yes.

- Limited overlap: no/yes (first, we used standard overlap; next, we used limited overlap).

Figure 6.11 shows two examples of discovered substructures in this use-case. Our domain expert evaluated the generated results focusing in the following issues: there is a modification for the population agglomerative criterion on the East side ("San Francisco", "El Alto Xonaca", and "Los Remedios" neighborhoods) and on the West side ("San José", "El Sagrario" and "El Carmen" neighborhoods) of the river. The "Mestizos" is the predominant ethnic group (24.5%); the "Spanish" is the next one (almost has the same percentage). This pattern may outline that the "Mestizos" ethnic group played the role of intermediator between the "Spanish" and "Undefined" groups on the West side, and between the "Spanish" and "Indígenas" on the East side.



a)



b)

Figure 6.11. Examples of discovered patterns in use-case "people around the river crossing Puebla downtown"

We needed to compare the previous results (using overlap) with the results using limited overlap, then, we used the same parameters as the previous test. We proposed to allow overlap only in vertices representing the ethnic group.

It is important to mention that, if our overlap label list has many elements, it could be more useful (concerning to processing time) to use standard overlap because of the overhead that results from the validation process. Remember that when we use the limited overlap feature, each time that an overlap among instances of a substructure is detected, the overlap is evaluated in order to know if it is allowed or not.

After comparing the results of using overlap and limited overlap, we found that the generated results were the same but not in processing time. Figure 6.12 shows the time comparison chart, standard overlap is shown in pink color (15,572 seconds) and limited overlap in yellow color (14,021 seconds).

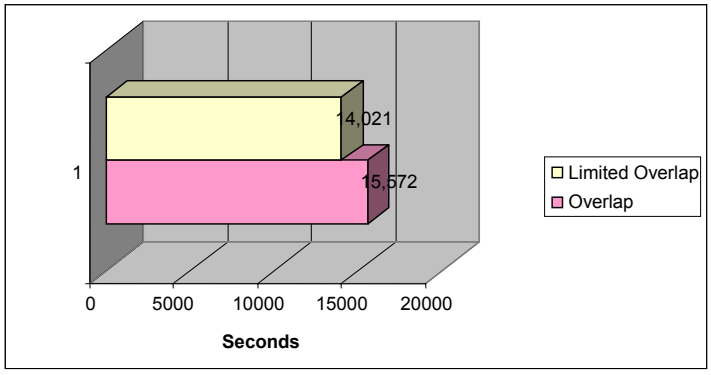


Figure 6.12. Processing time standard vs. limited overlap: use-case "people around the river crossing Puebla downtown".

The two use-cases presented in this section show the functionality of our proposal for modeling and mining spatial data using a graph-based representation. The discovered patterns in the population census from the year of 1777 were analyzed by a domain expert. Some of these patterns allow the user to validate facts already known. For instance, the predominant ethnic groups classified by parish, and population distribution according to the social status in the zone. But other patterns allow him to know implications, previously unknown, among spatial concepts and non-spatial attributes in the census. For instance, racial and economic interchange among people in the parishes, which are the common characteristics of population living along the borders of the river crossing downtown (on the West and East sides), social structure according to the ethnic group, common regularities among the family type and habitation space. Processing time comparison among standard and limited overlap features was other topic evaluated in these use-cases. We presented time processing comparison charts showing the time reduction gain obtained by using the new approach. We also show that by using limited overlap we can orient the search over substructures (patterns) containing elements that in our domain might represent relevant issues. For instance, in that time period the ethnic group represented a significant element to know characteristics about a family and their habitation space. Next section presents a use-case using a Popocatépetl volcano database. The objective in this illustrative use-case is to evaluate/compare the generated results by each proposed graph-based model.

6.2 Popocatépetl volcano

In chapter 4.3 we presented a preliminary use-case showing the applicability of our methodology using a Popocatépetl volcano database. This section presents an extended use-case employing the same database. First, we describe the study area, next the used method, and finally the generated results from the data mining phase.

As already mentioned, the database contains data related to several issues such as settlements, rivers, and evacuation roads in the zone. Figure 6.13 shows a fragment of the Popocatépetl volcano area. For the experiments we will use three spatial layers: roads (representing the roads in the area), rivers (representing the rivers in the area), and settlements (representing population areas). To illustrate the use-case we have delimited the study zone as shown in the figure.

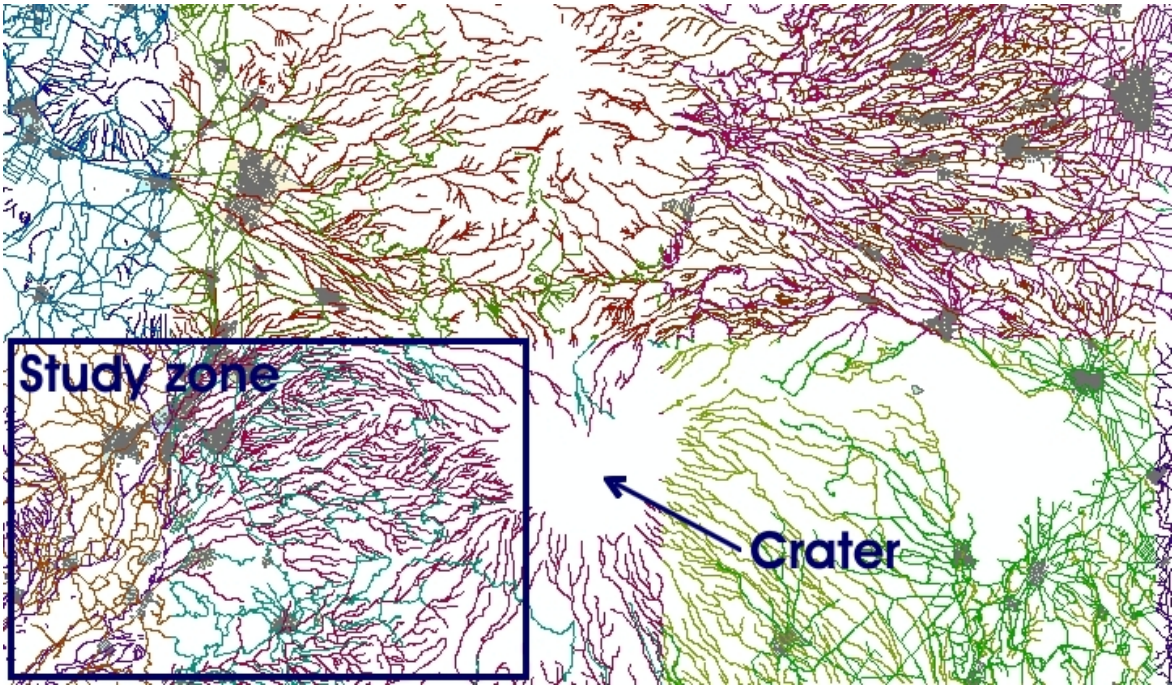


Figure 6.13. Popocatépetl volcano.

The experiments will focus in identifying relationships and characteristics shared among settlements, roads and rivers in the study zone. Suppose we want to know characteristics shared by these elements that can help us to implement/evaluate evacuation plans in case of a volcanic contingence". For example, characteristics of roads starting in or crossing a settlement, material used to build those roads and their current status (i.e. paved, unpaved), characteristics of the roads and rivers meeting a relationship (i.e. they cross, touch) in the zone, rivers near to settlements that in case of huge pluvial concentration might represent a potential risk.

6.2.2. Use-case: Popocatépetl

To illustrate the capabilities of our model we will use as our study zone that shown in Figure 6.14 (Southwest side of the volcano crater). The experiment is focused on discovering characteristics among roads, rivers, and settlements in the zone. However, the presentation of generated results is structured in the following way: discovered patterns among roads and rivers, roads and settlements, and rivers and settlements. The idea to choose this structure is only to show the different patterns that we can find among these elements.

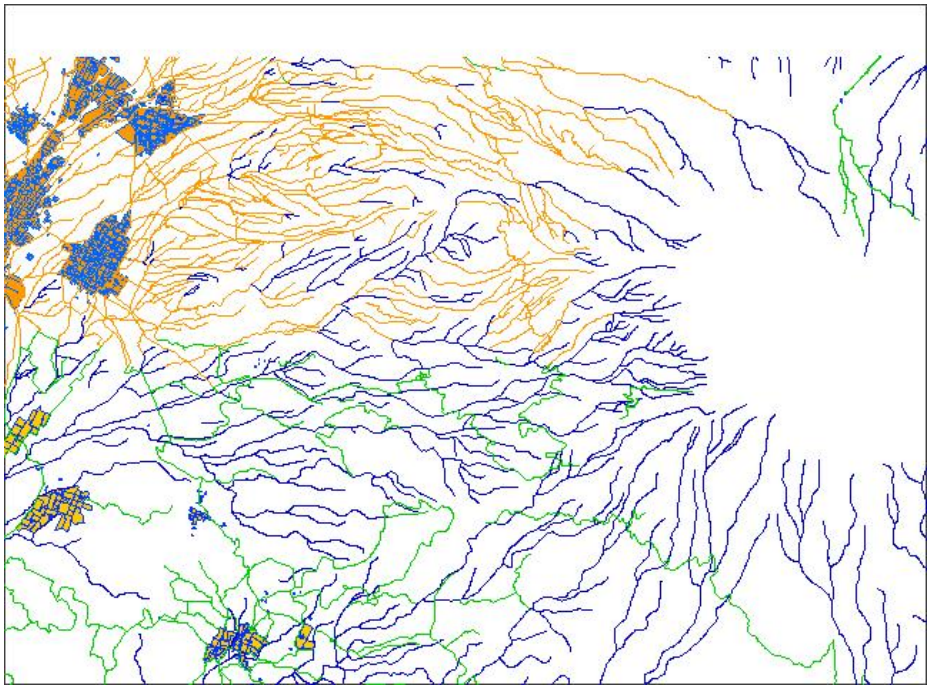


Figure 6.14. Popocatépetl volcano: study zone.

Therefore, we used our prototype system to select the river, settlement, and road spatial layers. The next step is to select the spatial relationships to be validated among the spatial objects under consideration. To develop our test, we take advantage of a special feature implemented in the Spatial Oracle module (our SDBMS system): Oracle has the capability to examine two geometry objects to determine their topological spatial relationship. Moreover, it is possible to indicate that the same SDBMS determines and returns the topological relationship that best matches the geometries.

So, to create our dataset, we first select the spatial layers to work with and then we use that feature to evaluate the relationships among the spatial objects. The experiments are implemented based on the following parameters:

- Spatial layers: *rivers*, *settlements* and *roads*.

- Spatial relation: topological relations supported by Oracle Spatial.
- Spatial graph-based model: all models.

The experiment was performed using the five proposed graph-based models. The objective was to evaluate the results generated by each of them. Additionally, we ran the test using no overlap, standard overlap and limited overlap. In the following figures we show the generated results in the experiment. In the figures the generated result by using no overlap is labeled as "*a)*", via standard overlap is labeled as "*b)*", and through limited overlap is labeled as "*c)*". In the case of limited overlap, we stated that Subdue allows overlap only for vertices representing roads in the zone since this element represents a primary item in our study (evaluation and implementation of population evacuation plans).

Since our intention with this experiment is to compare the results using the proposed graph-based models, we present as the most significant discovered pattern (by using no overlap, standard overlap and limited overlap), the one covering the following restrictions:

- Complete pattern. A pattern reporting at least two spatial objects (i.e. road and river), the spatial relation among them (i.e. touch), and some non-spatial attribute(s) (i.e. "road category unpaved" and "river category draining").
- Maximum number of reported instances. A pattern with the highest score of reported instances of a substructure.

The following subsections present the generated results using model 1 to 5. By each model we describe the discovered patterns according to the proposed structure: road and river,

road and settlement, and finally river and settlement. At the end of the section 6.2.2, we present comparison tables and conclusions of the generated results by each model.

6.2.2.1 Model #1 - base model

Road and River. Figure 6.15 shows the most significant discovered pattern between roads and rivers. The pattern describes a relationship among "road category unpaved overlapping a river category draining" in the zone. This pattern may be considered as an indicator of the number of roads that need to be supervised in case of a volcanic contingence since the material type they are built with, and because they cross rivers (the lecture may be done in inverse order) that in case of huge pluvial concentration may overflow and make roads useless. Subdue found by using no overlap 46 instances (second iteration) of the pattern; via standard overlap found 85 instances (first iteration), and through limited overlap also found 85 instances (second iteration). As we can see in the figure, standard and limited overlap found the same number of instances, but limited overlap required two iterations to find the same pattern. However, this fact does not mean that standard overlap is better than limited overlap because analyzing the overall processing time required by limited overlap to finish the substructure discovery phase we note that it was lower than the required by standard overlap.

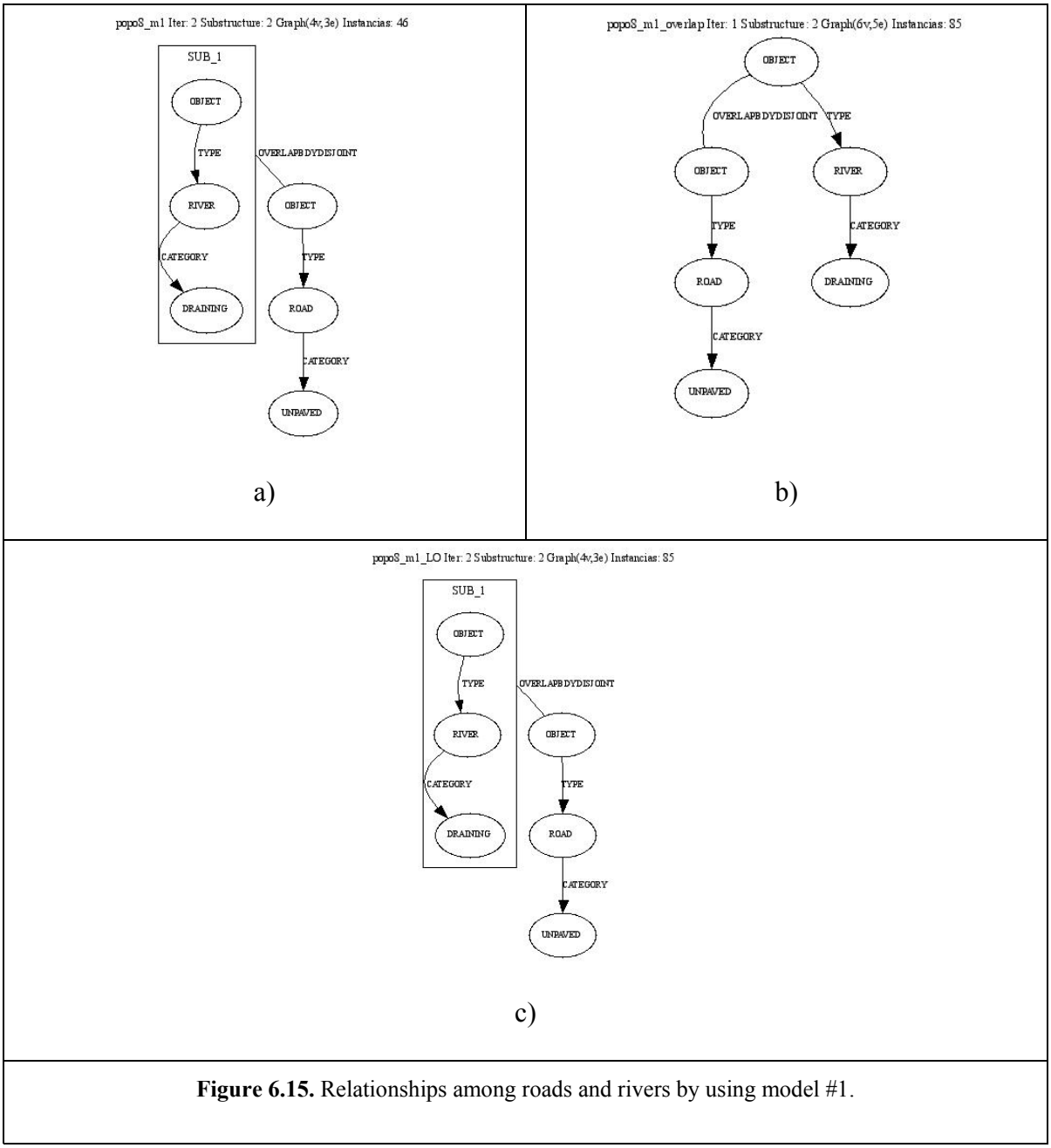


Figure 6.15. Relationships among roads and rivers by using model #1.

Road and Settlement. The most significant discovered pattern describes a relationship among "road category unpaved touching a settlement category construction" in the zone as shown in Figure 6.16. "Settlement category construction" represents in the Popocatépetl's settlement spatial layer inhabit areas with huge population, buildings and several constructions used to offer services to the people. If we assume that people may require to

be evacuated in case of an eruption and that the roads that will be used are unpaved then this situation may become a problem (i.e. a bottleneck, water and soil may become mud and this may make roads useless). For this experiment Subdue found via no overlap 6 instances (ninth iteration) of the pattern, through standard overlap 9 instances (fourth iteration), and by using limited overlap 8 instances (tenth iteration). In all cases Subdue was able to discover the same pattern; the difference was the number of computed iterations.

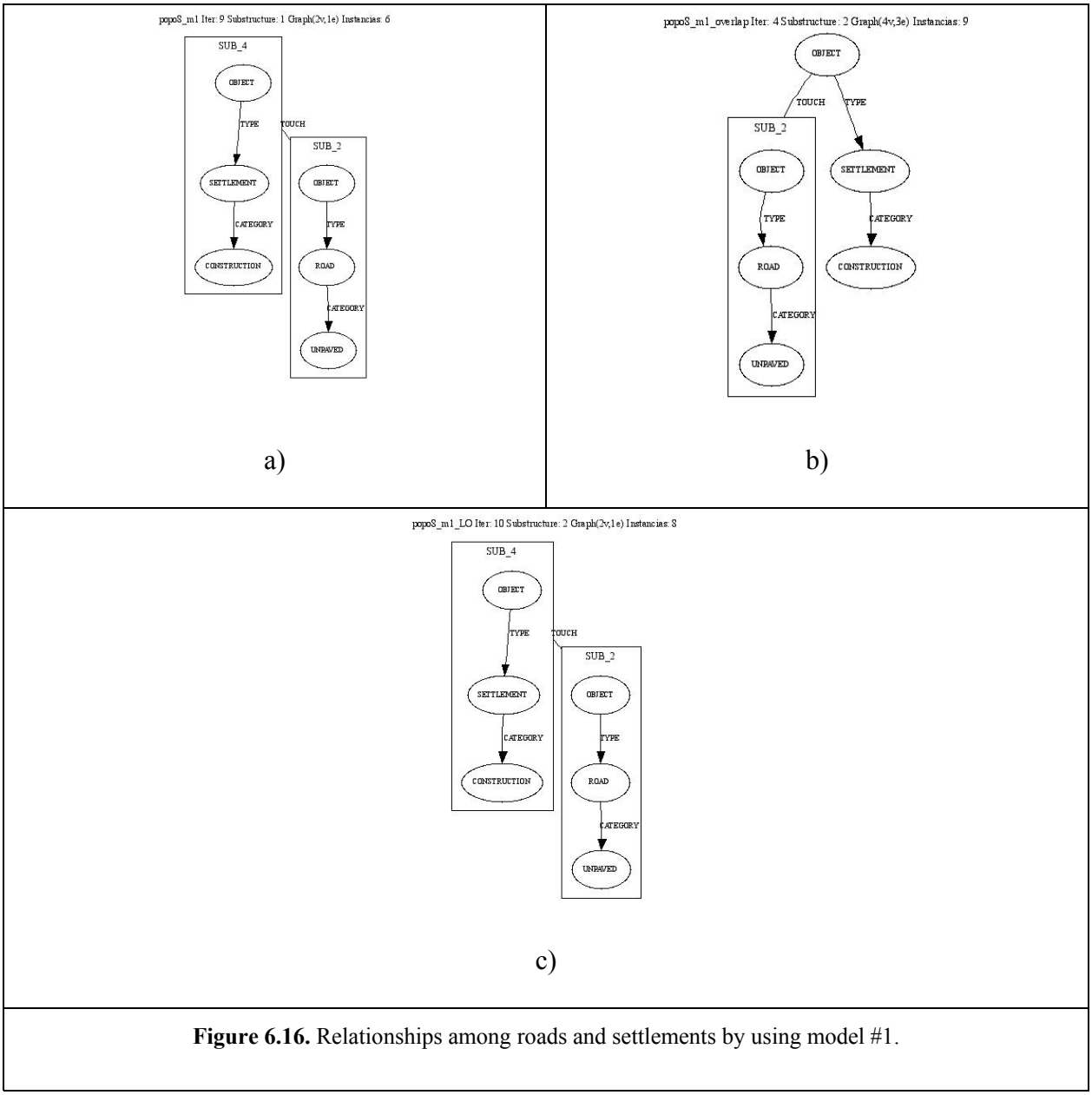


Figure 6.16. Relationships among roads and settlements by using model #1.

River and Settlement. Figure 6.17 shows the most significant discovered pattern for these spatial objects. It describes a relationship among “river category draining crossing a settlement category either (a) block or (b)(c) construction” in the zone. “Settlement category block” represents in the Popocatépetl´s settlement spatial layer inhabit areas but with little population, in fact there exist several uninhabited areas, few buildings and constructions. The pattern may be used to identify potential inundation zones because it represents rivers close to (may be some of them crossing) areas inhabited by people. Through no overlap Subdue found 5 instances (twelfth iteration), by using standard overlap found 5 instances (eighth iteration), and via limited overlap also found 5 instances (eighth iteration). Subdue discovered the same pattern in the three cases, however, by using standard overlap and limited overlap the lecture of the pattern is simpler.

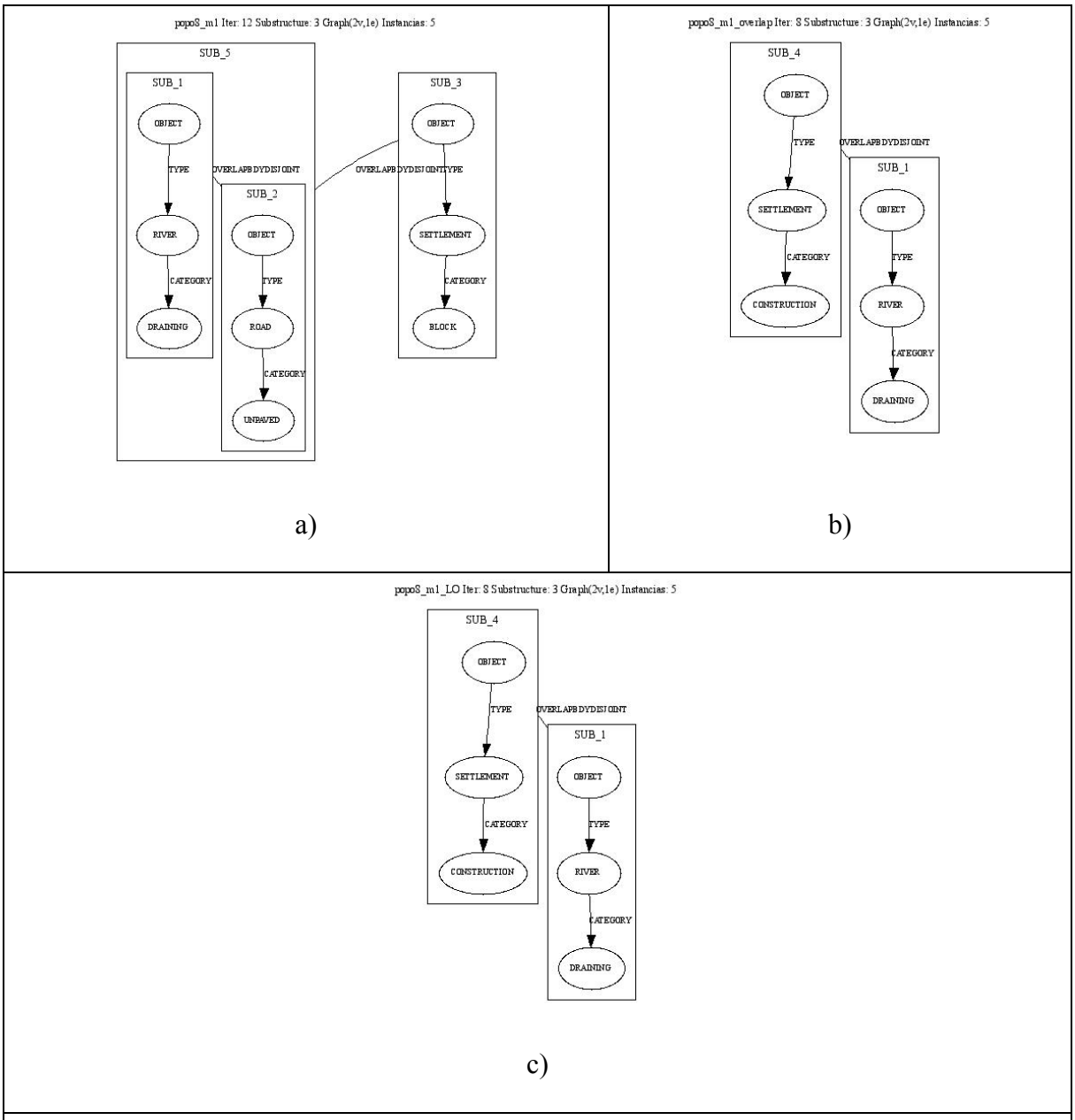


Figure 6.17. Relationships among rivers and settlements by using model #1.

The previous experiment was done using model 1, now we perform the same experiment using model 2 to 5. We will be focused to compare the same discovered patterns for the three "object-object" structures (i.e. road-river, road-settlement, and river-settlement). The

generated results show in the figures follow the same organization: by using no overlap is labeled as "*a)*", via limited overlap is labeled as "*b)*", and through limited overlap is labeled as "*c)*". The most significant differences between the generated results, in this experiment, based on the number of reported instances and the number of iterations needed to discover the pattern. More iterations means more processing time to discover the pattern.

6.2.2.2 Model #2 - single replication of relations types, complete information

Road and River. By means of model #2 Subdue discovered the pattern shown in Figure 6.18. Subdue found by using no overlap 41 instances (second iteration) of the pattern, via standard overlap found 85 (third iteration), and through limited overlap found 64 (second iteration). All cases reported "road category unpaved and river category draining" as the predominant spatial objects.

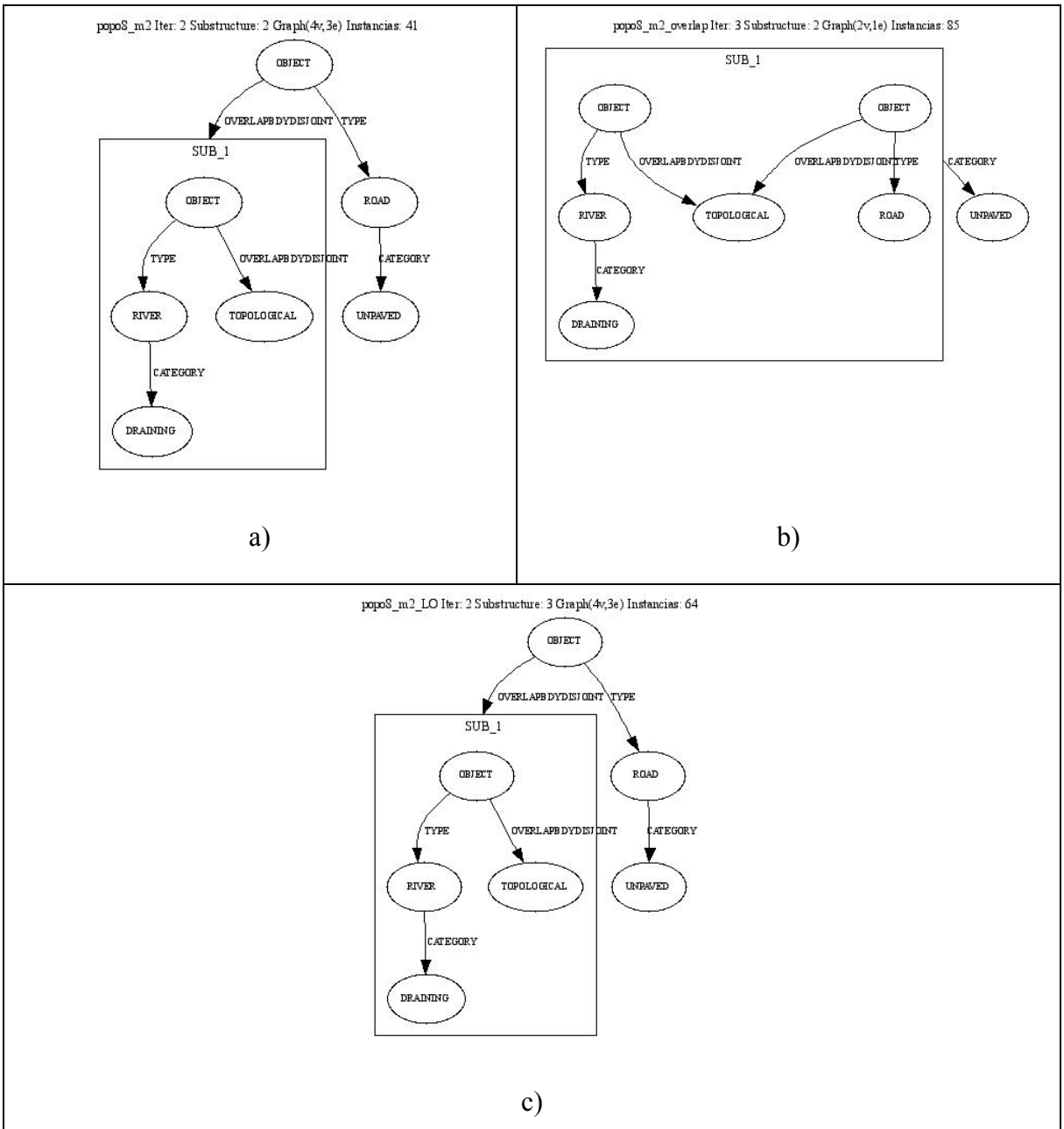


Figure 6.18. Relationships among roads and rivers by using model #2.

Road and Settlement. For these spatial objects Subdue was able to discover, by means of model #2, the pattern shown in Figure 6.19. Via no overlap Subdue found 5 instances

(fourteenth iteration) of the pattern, through standard overlap found 8 (sixth iteration), and by using limited overlap found 7 (tenth iteration). No overlap reported "settlement category block" whereas standard and limited overlap reported more instances of "settlement category construction".

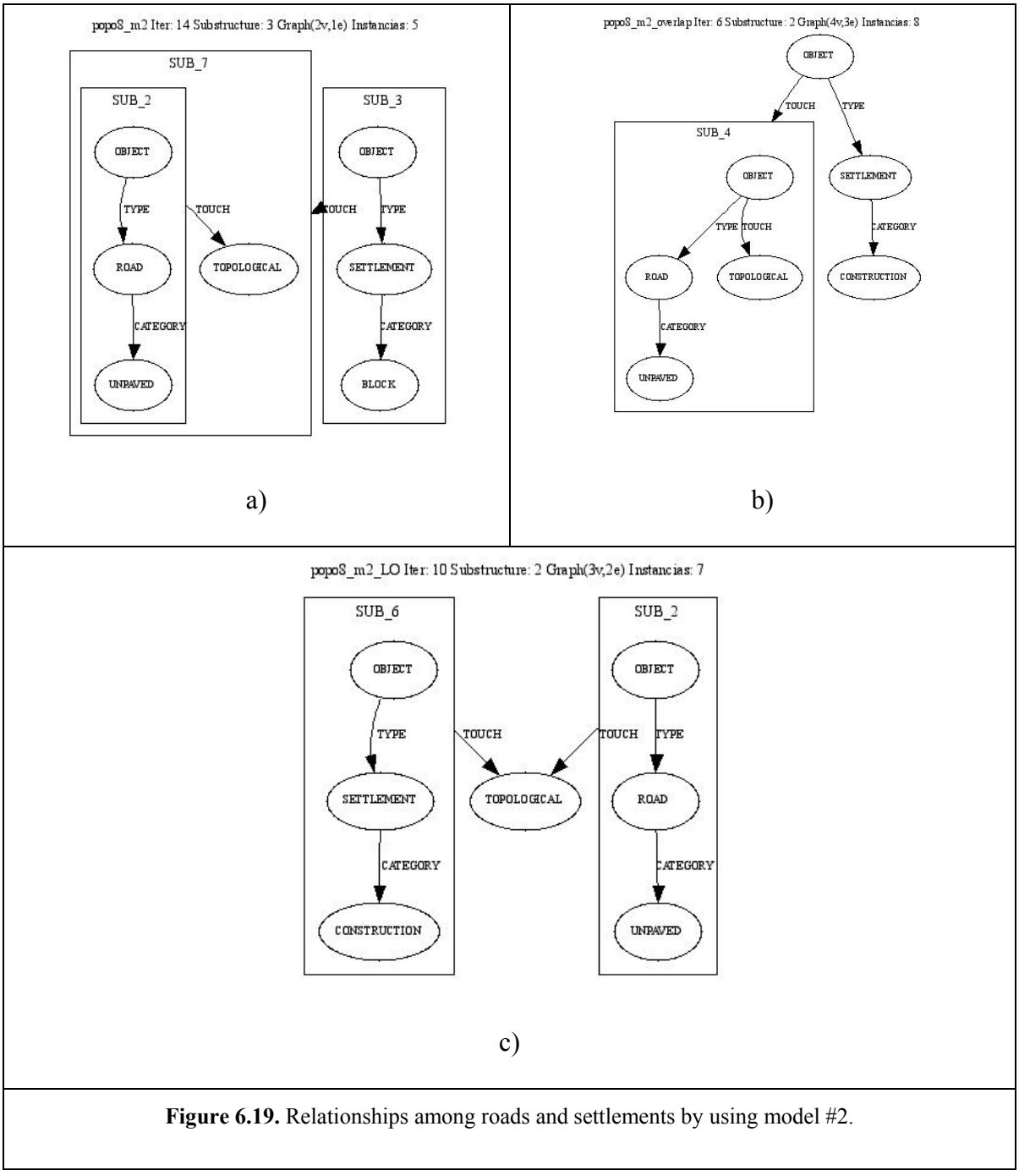


Figure 6.19. Relationships among roads and settlements by using model #2.

River and Settlement. The pattern discovered, by means of model #2, for these objects is shown in Figure 6.20. Through no overlap Subdue found 5 instances (sixteenth iteration) of the pattern, by using standard overlap found 10 (seventh iteration), and via limited overlap found 5 (fourteenth iteration). No overlap and limited overlap reported "settlement category construction" whereas limited overlap reported more instances of "settlement category block".

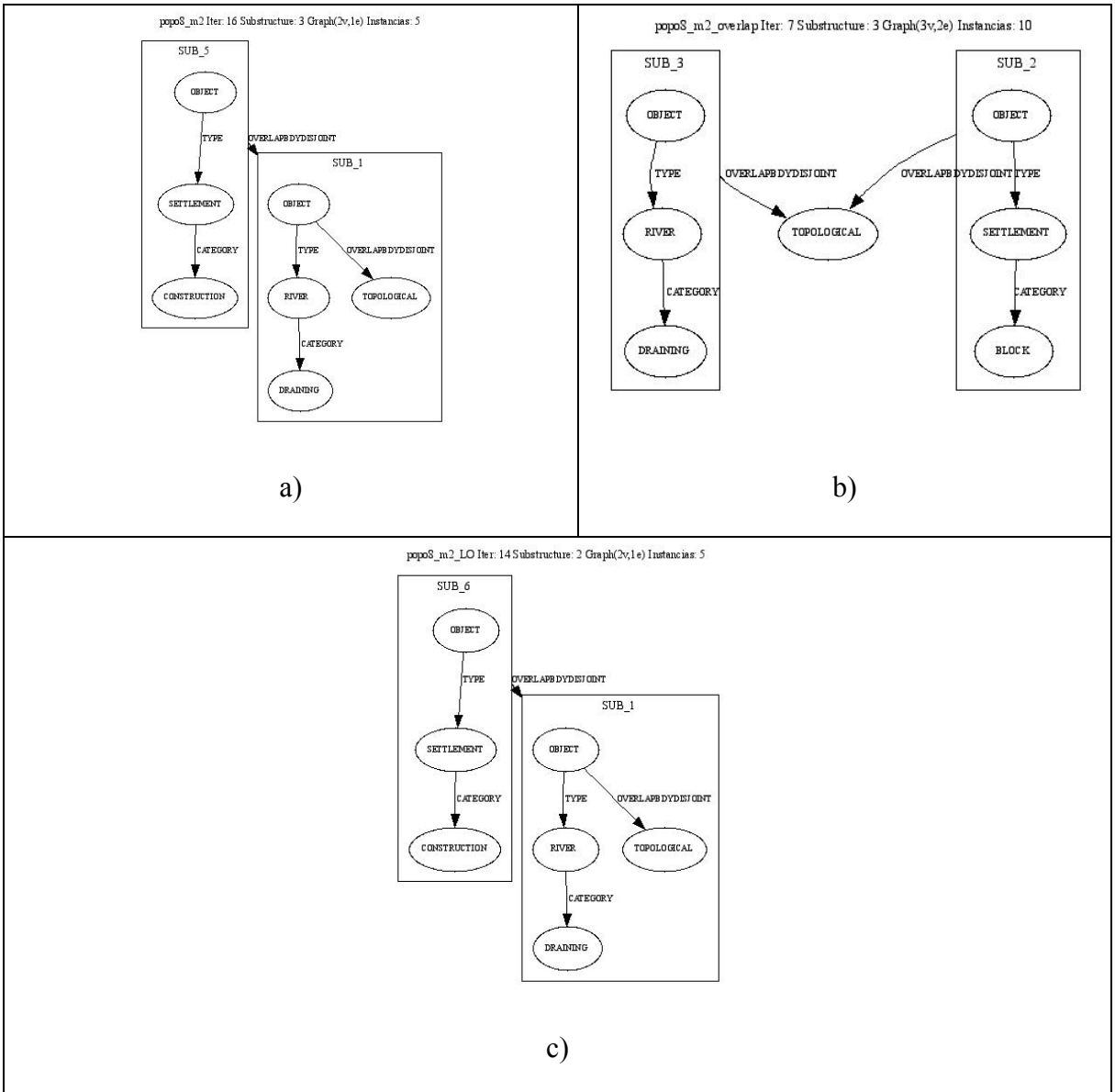


Figure 6.20. Relationships among rivers and settlements by using model #2.

6.2.2.3 Model #3 - double replication of relations types, no complete information

Road and River. The pattern discovered, by means of model #3, for these objects is shown in Figure 6.21. Through no overlap Subdue found 39 instances (second iteration) of the pattern, by using standard overlap found 85 (second iteration), and via limited overlap found 34 (ninth iteration). In all cases Subdue reported as the predominant spatial objects "road category unpaved and river category draining".

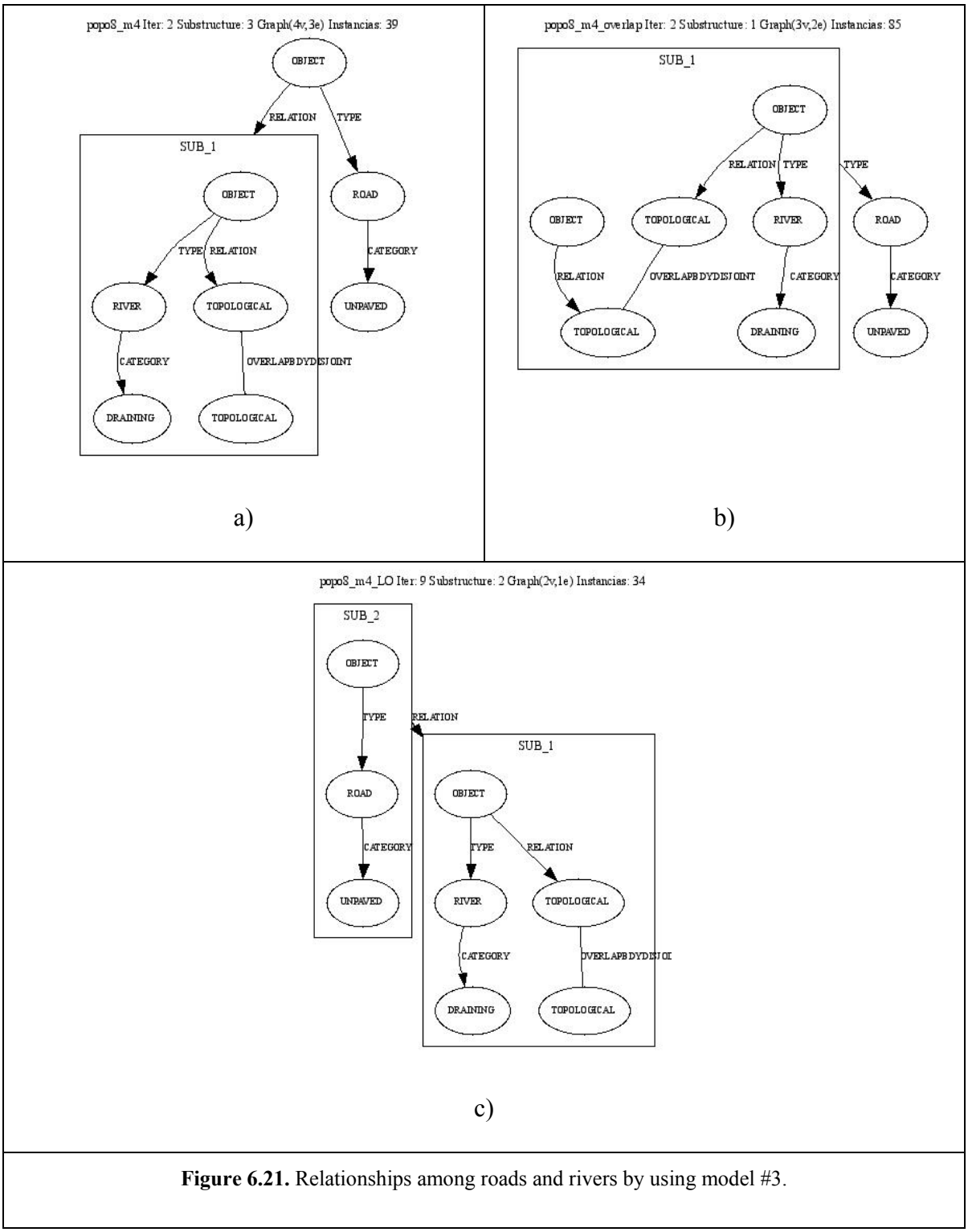


Figure 6.21. Relationships among roads and rivers by using model #3.

Road and Settlement. By means of model #3 Subdue discovered the pattern shown in Figure 6.22. Subdue found by using no overlap 4 instances (fifteenth iteration) of the pattern, via standard overlap found 8 (sixth iteration), and through limited overlap found 5 (thirteenth iteration). No overlap and limited overlap reported "settlement category block" as the predominant spatial object whereas standard overlap reported "settlement category construction".

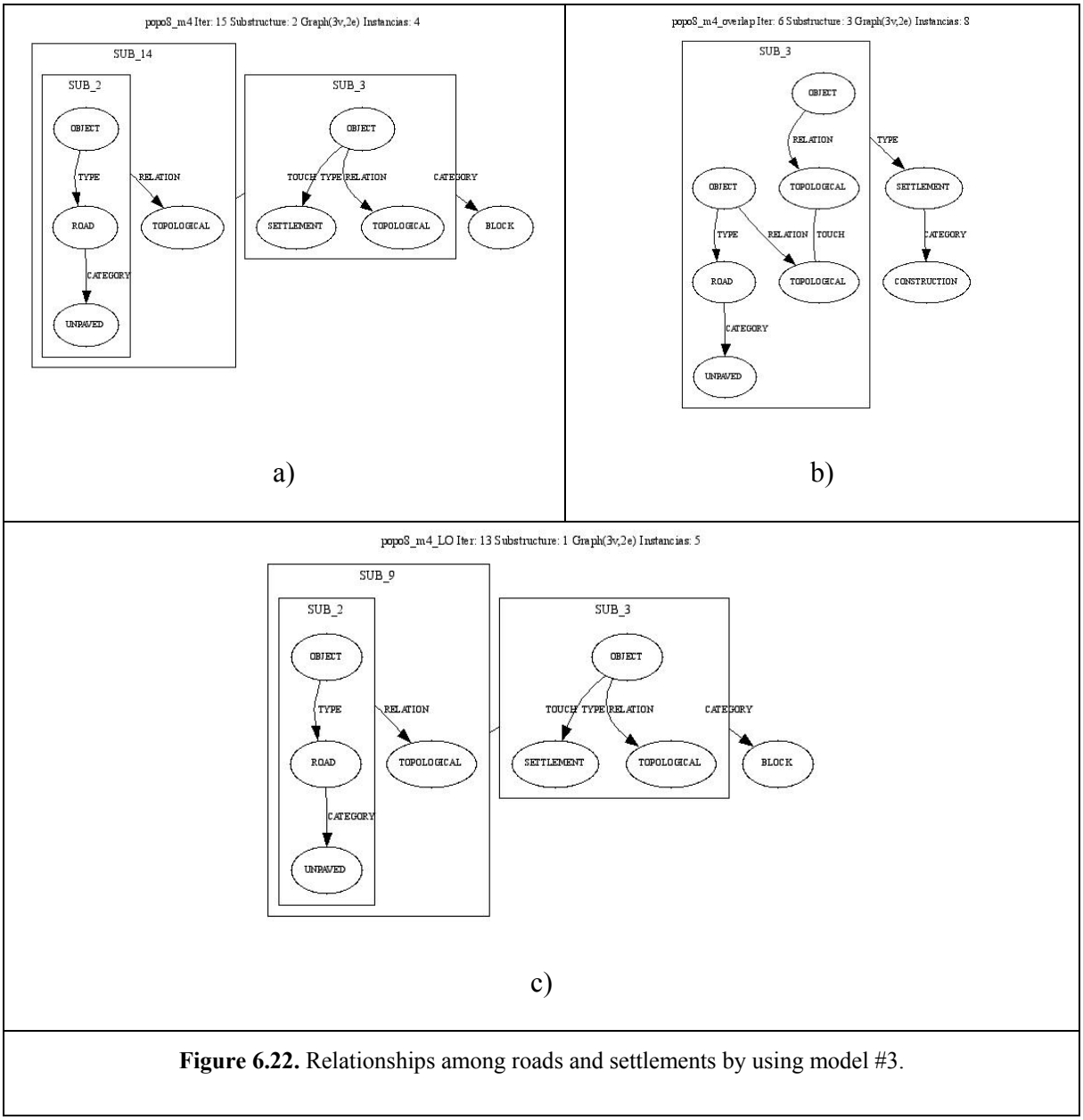


Figure 6.22. Relationships among roads and settlements by using model #3.

River and Settlement. For these spatial objects Subdue was able to discover, by means of model #3, the pattern shown in Figure 6.23. Via no overlap Subdue found 5 instances (twelfth iteration) of the pattern, through standard overlap found 19 (fourth iteration), and by using limited overlap found 5 (tenth iteration). No overlap and limited overlap reported less instances of "settlement category construction" whereas standard overlap reported more instances of "settlement category block".

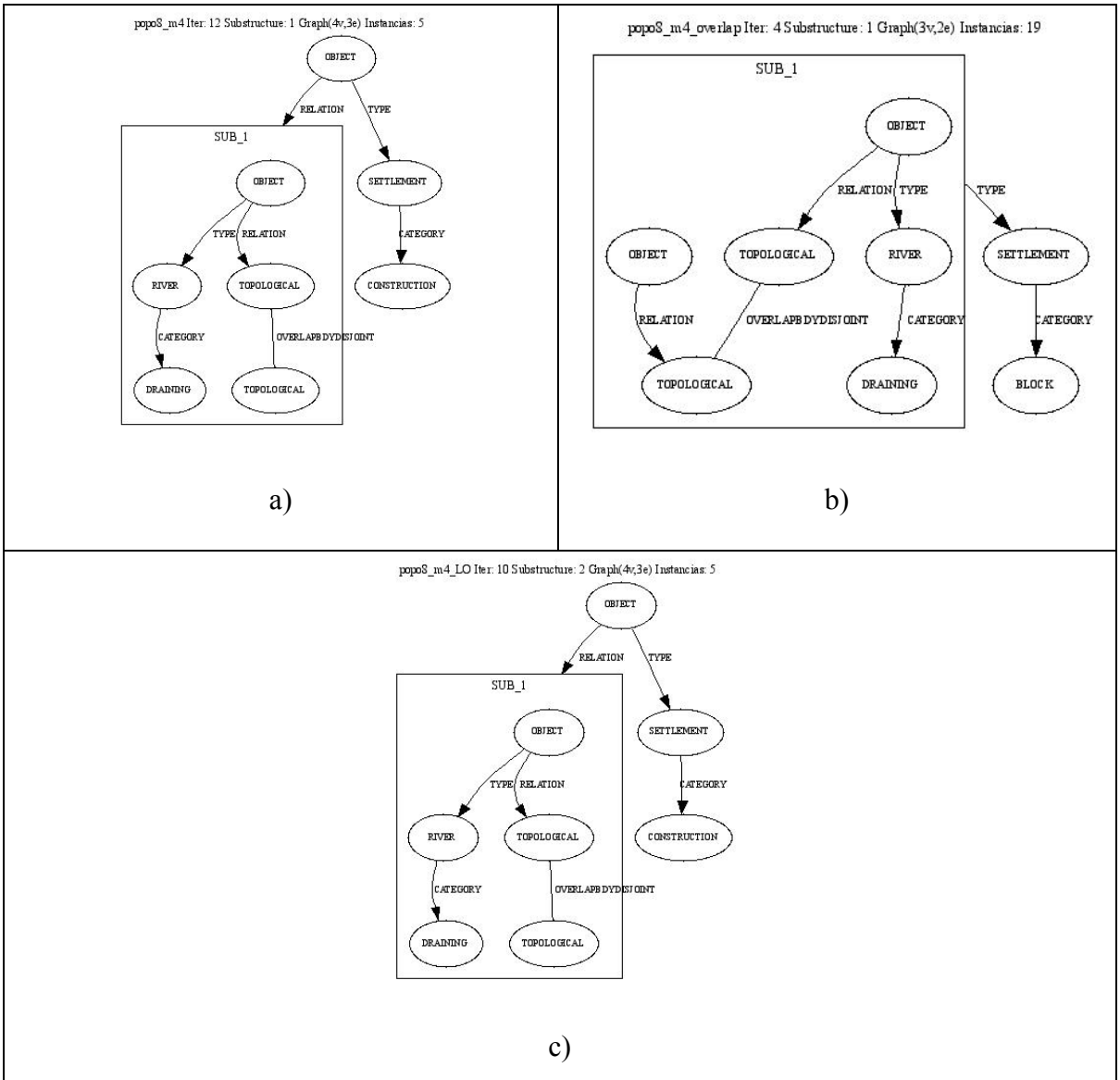


Figure 6.23. Relationships among rivers and settlements by using model #3.

6.2.2.4. Model #4 - single replication of relations types, no complete information

Road and River. For these spatial objects Subdue was able to discover, by means of model #4, the pattern shown in Figure 6.24. Via no overlap Subdue found 39 instances (second iteration) of the pattern, through standard overlap found 85 (first iteration), and by using limited overlap found 60 (second iteration). All cases reported as the predominant spatial objects “road category unpaved and river category draining”.

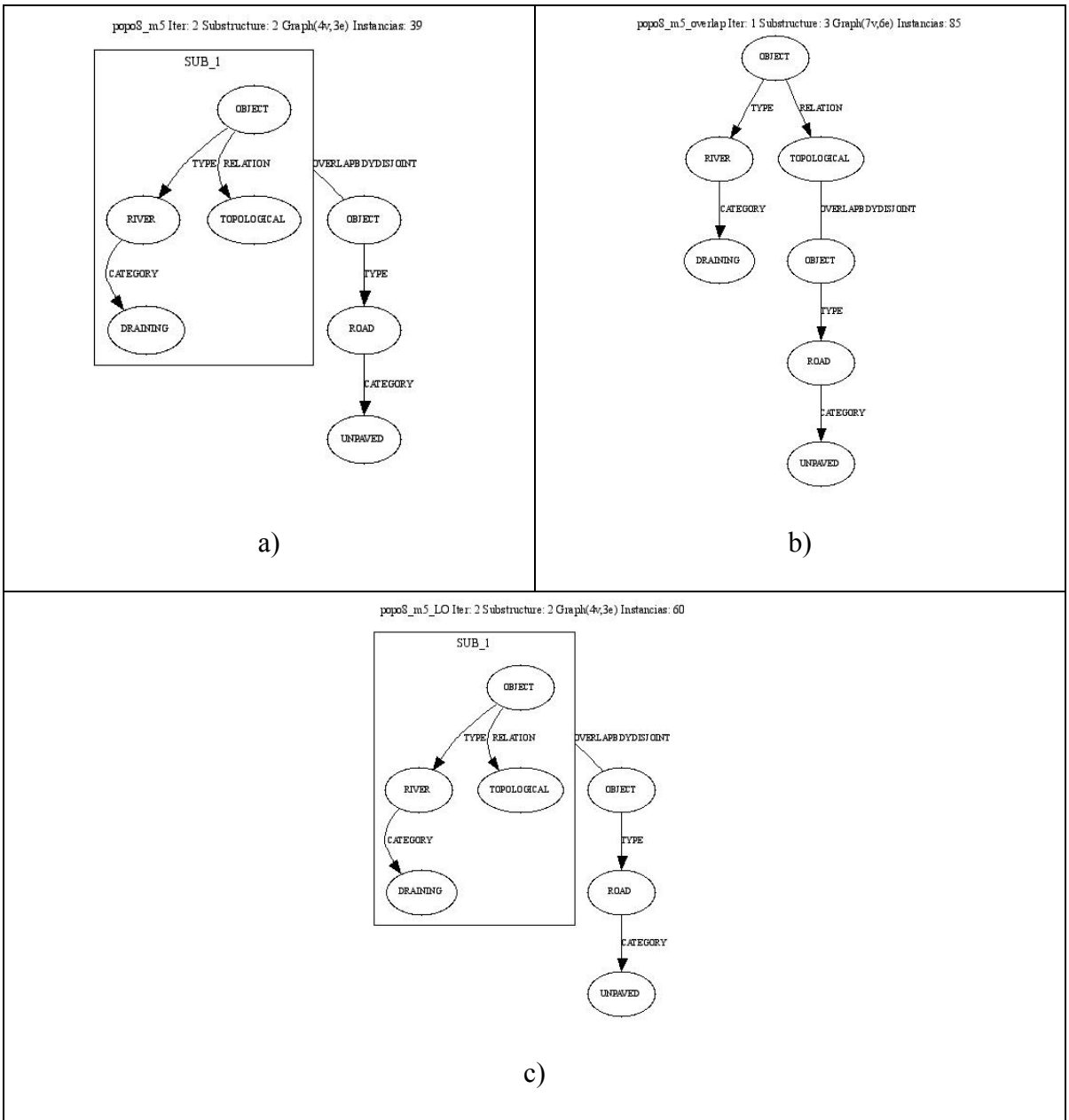


Figure 6.24. Relationships among roads and rivers by using model #4.

Road and Settlement. The pattern discovered, by means of model #4, for these objects is shown in Figure 6.25. Through no overlap Subdue found 6 instances (twelfth iteration) of the pattern, by using standard overlap found 8 (tenth iteration), and via limited overlap

found 8 (seventh iteration). The representative spatial objects are "road category unpaved and settlement category construction" in all cases.

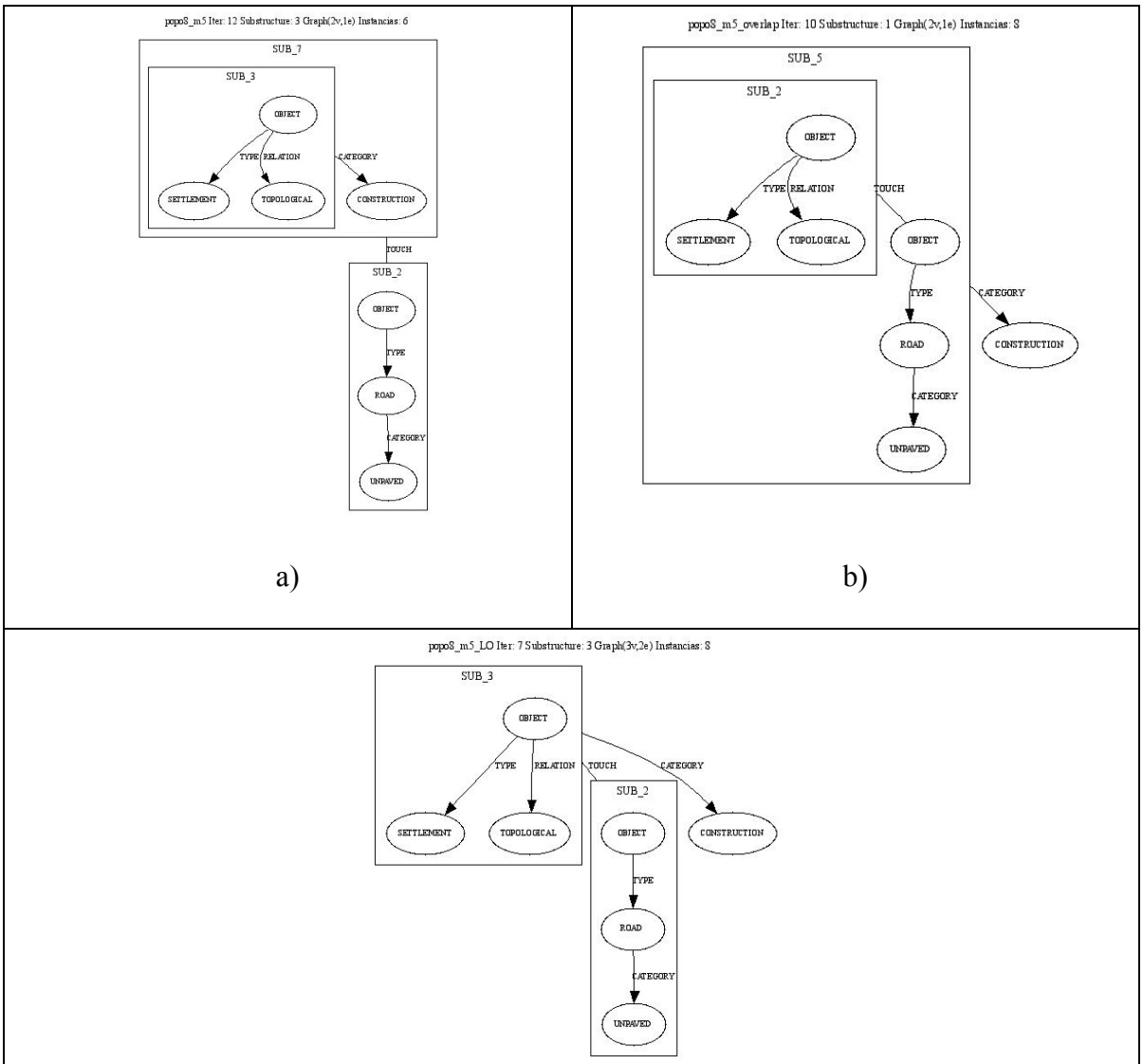


Figure 6.25. Relationships among roads and settlements by using model #4.

River and Settlement. By means of model #4 Subdue discovered the pattern shown in Figure 6.26. Subdue found by using no overlap 5 instances (sixth iteration) of the pattern,

via standard overlap found 10 (sixth iteration), and through limited overlap found 5 (eleventh iteration). No overlap and limited overlap reported less instances of "settlement category construction" whereas standard overlap reported more instances of "settlement category block".

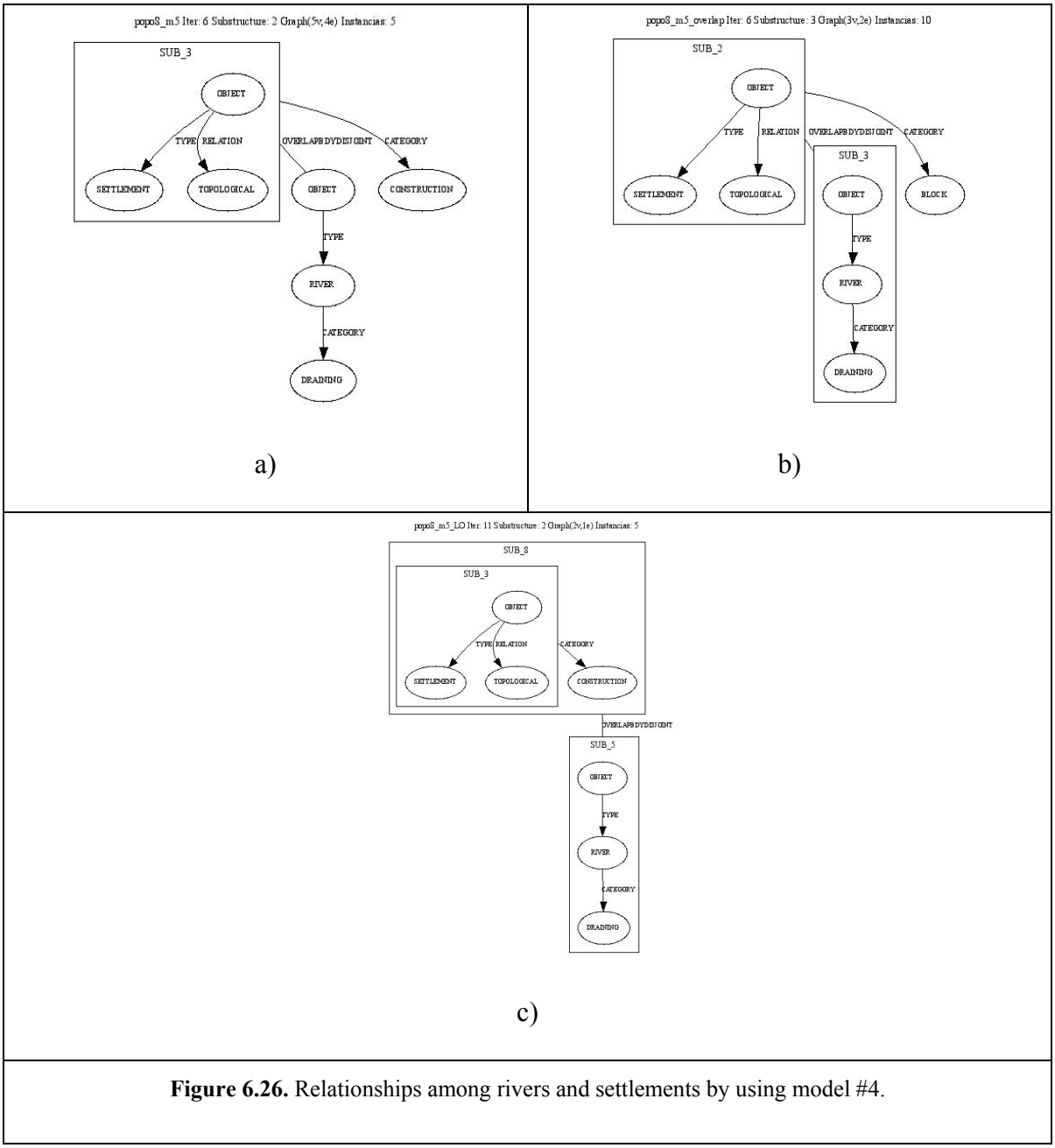


Figure 6.26. Relationships among rivers and settlements by using model #4.

6.2.2.5 Model #5 - double replication of relations types, complete information

Road and River. The pattern discovered, by means of model #5, for these objects is shown in Figure 6.27. Through no overlap Subdue found 45 instances (fifth iteration) of the pattern, by using standard overlap found 85 (first iteration), and via limited overlap found 45 (fifth iteration). Subdue reported in all cases as the representative spatial objects "road category unpaved and river category draining".

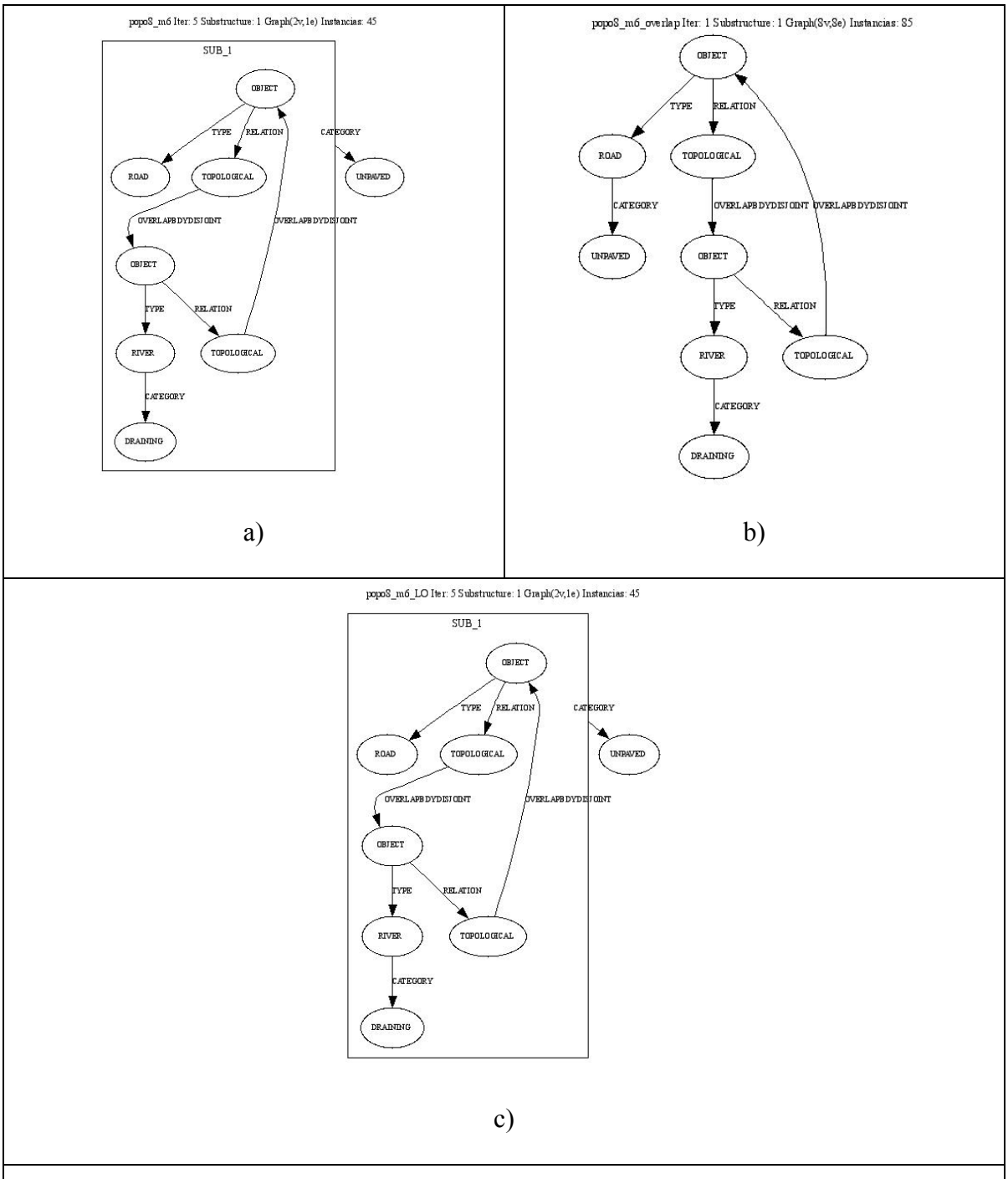


Figure 6.27. Relationships among roads and rivers by using model #5.

Road and Settlement. By means of model #5 Subdue discovered the pattern shown in Figure 6.28. Subdue found by using no overlap 6 instances (seventh iteration) of the pattern, via standard overlap did not find a complete pattern (in the figure, the category of the settlement is not reported), and through limited overlap found 7 (seventh iteration). No overlap and limited overlap reported as the representative spatial objects "road category unpaved and settlement category construction".

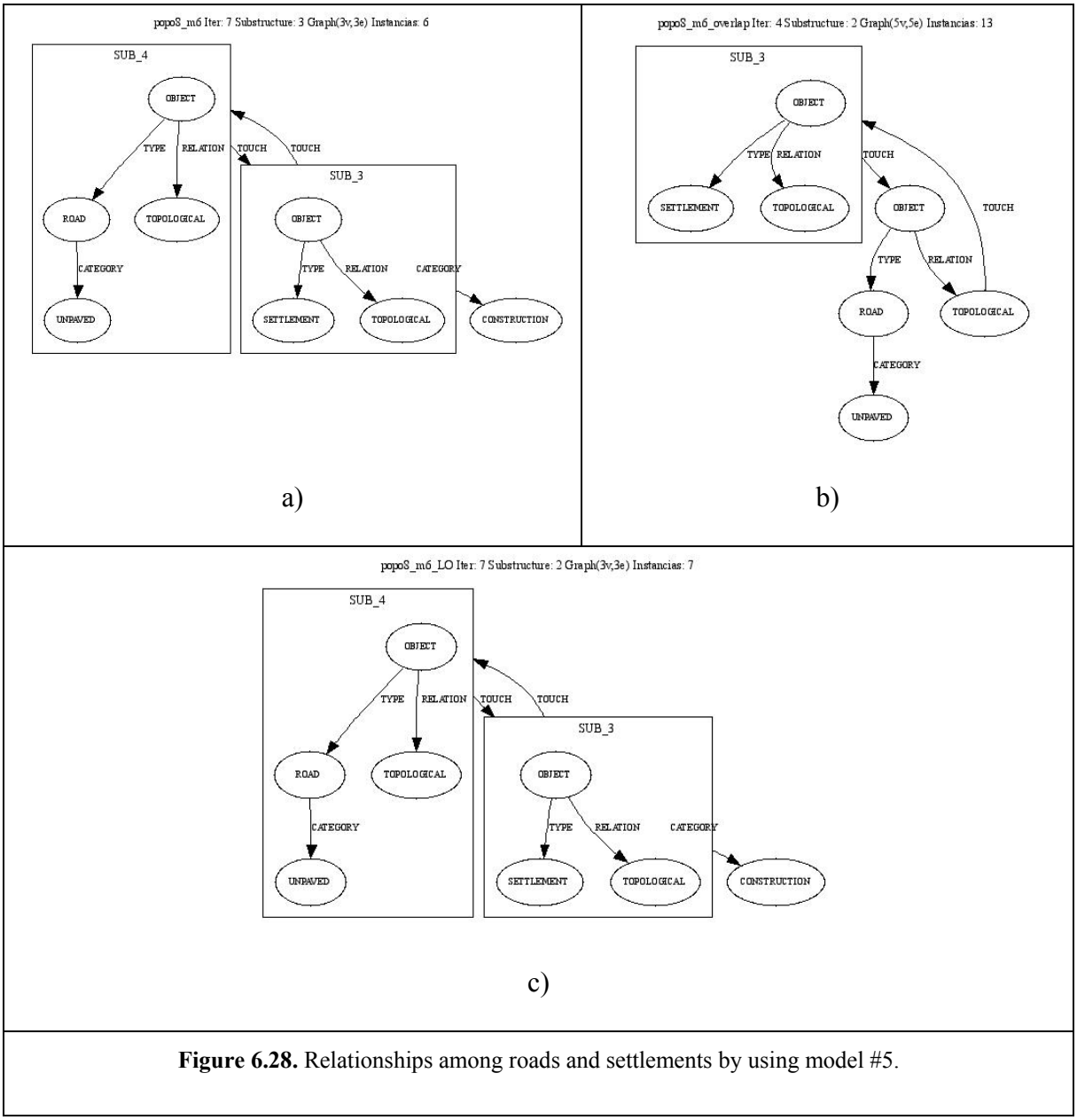


Figure 6.28. Relationships among roads and settlements by using model #5.

River and Settlement. For these spatial objects Subdue was able to discover, by means of model #5, the pattern shown in Figure 6.29. Via no overlap Subdue found 5 instances (thirteenth iteration) of the pattern, through standard overlap found 5 (sixth iteration), and by using limited overlap found 5 (tenth iteration). Subdue reported in all cases "river category draining and settlement category construction" as the predominant spatial objects.

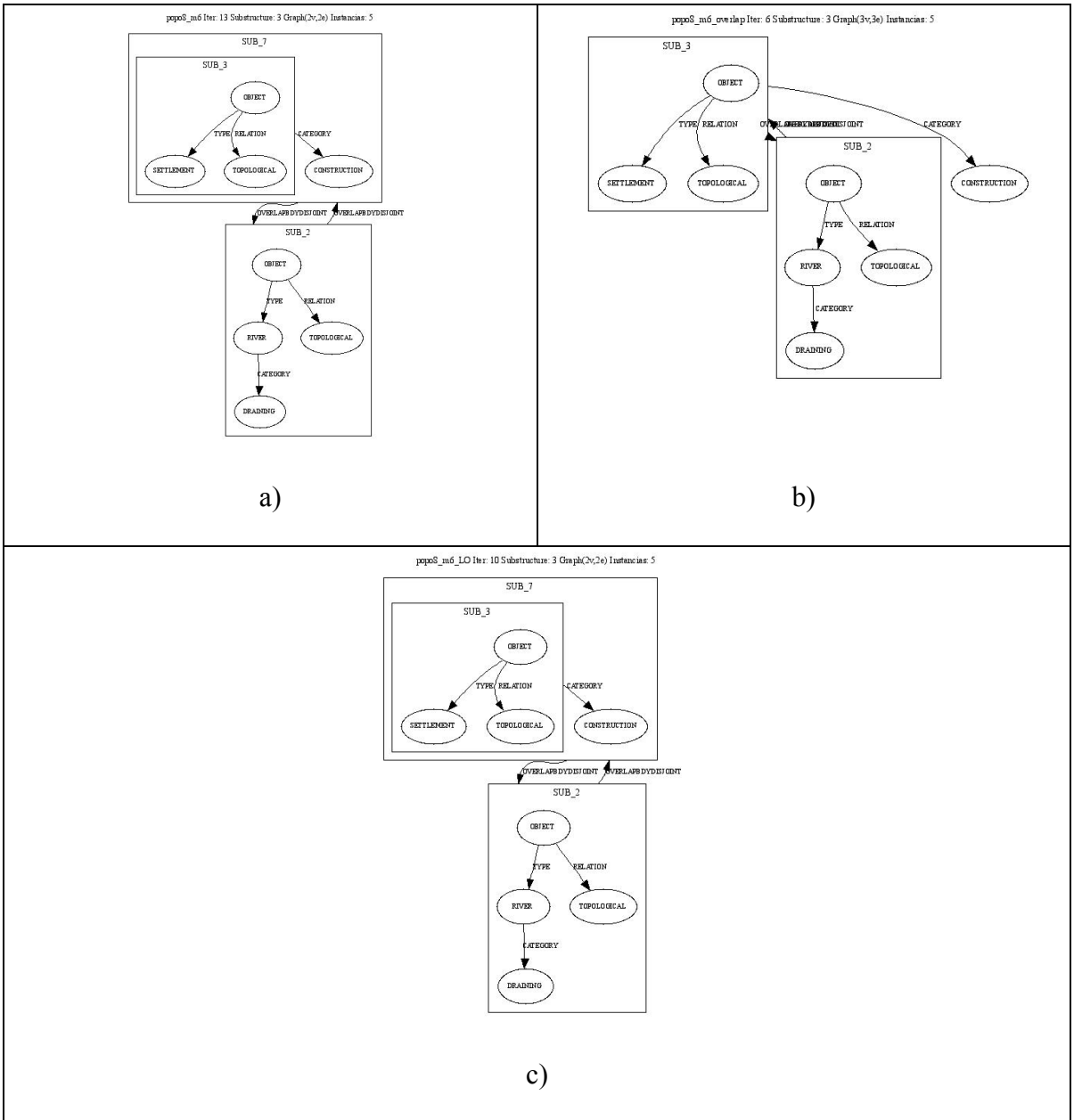


Figure 6.29. Relationships among rivers and settlements by using model #5.

Table 6.1 presents a comparison, by model, among the number of discovered instances/iteration need to discover them and the overlap features. For example, by using model #1, Subdue found 46 instances (in second iteration) of a "complete" pattern (according to our definition for reporting a complete pattern) involving the spatial objects road-river via no overlap. Higher score means a model allowing us to discover more instances of a substructure. Remember Subdue reported as the best pattern (by iteration) a substructure with the highest score of discovered instances of that substructure. This comparison is reported by each "object-object" structure.

Note: NO (No overlap), SO (Standard overlap), LO (Limited overlap).

	Model #1			**Model #2**			**Model #3**			**Model #4**			**Model #5**		
	NO	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**
Road-River															
Instances	46	85	85	41	85	64	39	85	34	39	85	60	45	85	45
Iterations	2	1	2	2	3	2	2	2	9	2	1	2	5	1	5
Road-Settlement															
Instances	6	9	8	5	8	7	4	8	5	6	8	8	6	0	7
Iterations	9	4	10	14	6	10	15	6	13	12	10	7	7	0	7
River-Settlement															
Instances	5	5	5	5	10	5	5	19	5	5	10	5	5	5	5
Iterations	12	8	8	16	7	14	12	4	10	6	6	11	13	6	10

Table 6.1. Instances/iterations by each graph-based model: Popocatépetl use-case.

Table 6.2 presents a comparison of maximum/minimum discovered instances by overlap features. A model with the highest score is better since it allows discovering more instances of a substructure (patterns). The comparison is presented by each "object-object" structure. For example, in the structure road-river, model #1 reported 46 discovered instances in the second iteration (the highest score). See Table 6.1 for details.

	Maximum	**Minimum**
Road and River		
No overlap	model #1 (second iteration)	models #3 and #4 (second iteration)
Overlap	models #1, #4 and #5 (first iteration)	model #2 (third iteration)
Limited overlap	model #1 (second iteration)	model #3 (ninth iteration)
Road and Settlement		
No overlap	model #5 (seventh iteration)	model #3 (fifteenth iteration)
Overlap	model #1 (fourth iteration)	model #5 (no completed pattern)
Limited overlap	model #4 (seventh iteration)	model #3 (thirteenth iteration)
River and Settlement		
No overlap	model #4 (sixth iteration)	model #2 (sixteenth iteration).
Overlap	model #3 (fourth iteration)	model #1 (eighth iteration).
Limited overlap	model #1 (eighth iteration)	model #2 (fourteenth iteration)

Table 6.2. Max/Min of discovered instances by "object-object"/overlap feature.

Table 6.3 presents a comparison among the average of discovered instances by model. Higher score means a model allowing us to discover more instances of a substructure (patterns). Each value represents the average of discovered substructures by using no overlap, standard overlap and limited overlap. The comparison is reported by each "object-object" structure.

	Model #1	Model #2	Model #3	Model #4	Model #5
Road and River	72.0	63.3	52.7	61.3	58.3
Road and Settlement	7.7	6.7	5.7	7.3	4.3
River and Settlement	5.0	6.7	9.7	6.7	5.0

Table 6.3. Average of discovered instances by model/"object-object".

Table 6.4 presents a comparison among the average of discovered instances by model. Higher score means a model allowing us to discover more instances of a substructure (patterns). The comparison is reported by each overlap feature.

Model #1			Model #2			Model #3			Model #4			Model #5		
NO	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**	**NO**	**SO**	**LO**
19.0	33.0	32.7	17.0	34.3	25.3	16.0	37.3	14.7	16.7	34.3	24.3	18.7	30.0	19.0

Table 6.4. Average of discovered instances by model/overlap feature.

Table 6.5 presents a final comparison among the average of discovered instances by model. We can see in the table that model #1 reported the highest score of discovered instances (according to our parameters for reporting complete instances) in this illustrative Popocatépetl use-case. The following ones are model #2 and model #4 respectively.

Model #1	Model #2	Model #3	Model #4	Model #5
28.2	25.6	22.7	25.1	22.6

Table 6.5. Average of discovered instances by model.

6.3 Conclusion

In this chapter we have presented three illustrative use-cases of our proposal for modeling and mining spatial data. The test contexts were two spatial databases, the first one related to a population census from the year of 1777 in Puebla downtown, and the second one related to the Popocatépetl volcano.

The use-cases developed using the population census database were focused to exemplify the graph-based model to represent together spatial data, non-spatial data and spatial relations, the new limited overlap feature implemented in the Subdue system (processing time and specialized overlapping pattern oriented search), and the generated results by the mining phase. We have presented in each use-case evaluations performed by the domain expert over the discovered patterns.

The use-case developed using the Popocatépetl database was focused to compare/evaluate the generated results by each proposed graph-based model. The tests were implemented upon the supposition of evacuation plans implementation in case of volcanic contingences. We presented comparison tables describing which model(s) allow(s) to discover more instances of a substructure via no overlap, standard overlap, and limited overlap features. Subdue reported as the best pattern (by iteration) a substructure with the highest score of discovered instances of that substructure.

Next chapter presents conclusion about our research work and final remarks.

Chapter 7

CONCLUSIONS

The continuous interaction among people and their natural home, the earth planet, generates everyday new requirements associated to spatial data. For example, urban analysis, natural risks prevention, space exploration, contamination in oceans, and reforestation of lands just to mention some of them. Spatial data mining involves the integration of methods from different scientific fields which help us, by means of data analysis and discovery algorithms produce a particular enumeration of patterns from spatial data.

In chapter 2 we presented several approaches developed for mining spatial data (i.e. generalization-based methods, clustering, spatial associations, approximation and aggregation, mining in image databases, spatial classification, and spatial trend detection). However, our argumentation about those approaches was that they do not consider all the elements found in a spatial database (spatial data, non-spatial data and spatial relations among the spatial objects) in an extended way. We proposed in this dissertation a new approach based on graphs. Our feeling is that if we are able to represent those elements as a unique dataset and if we are also able to mine them as a whole, then, we might be able to get patterns that might contain both types of data and spatial relationships enhancing the

quality of the results since the generated pattern(s) would describe (a) spatial object(s) meeting (a) spatial relation(s) with other spatial object(s) and which is(are) that(those) relationship(s). We proposed to use a graph-based representation since it provides the desirable flexibility to describe these elements and their relations together.

As mentioned, in our model spatial relations among spatial objects are included since a significant characteristic of spatial data is the influence of the neighbors of an object may have on the object itself. In the model we included the representation of three types of spatial relations.

Derived from the general graph-based schema we have proposed five operative models. Three aspects define the characteristics of a graph created from those models: first, the representation of equivalent spatial relations (the relation *A* touch *B* can be represented by two directed edges, $A \rightarrow B$ and $B \rightarrow A$, or by one undirected edge, $A \text{—} B$; we used the second approach). Second, the representation of symmetric spatial relations (the relation *A* North_of *B* implies a relation *B* South_of *A*, some models represent only the first relation and other both relations). The third aspect is the model itself. Our intention is to represent the spatial objects and their relation as much as possible but also considering a balance among the representative of the data and the complexity of the created graph. This last issue has a major importance since the closer relation among the complexity of the created graph and the mining phase. For examples, huge graphs may require more computational resources than small graphs, but by creating small graphs we may loss data representativeness and perhaps we may not to discover significant patterns.

As component of our methodology for mining spatial data using a graph-based approach, we used the Subdue system as our mining tool. The overlap feature plays a relevant role in the Subdue's substructure discovering system. But, as we described, it is implemented in an orthodox way: allows overlap among any instances of a substructure or allows overlap among all instances of a substructure. The first option is better regarding processing time, but the second one may discover more instances of a substructure (pattern). Both cases do not give to the user the capacity to specify the set of vertices allowed for overlap.

Therefore, we proposed a new overlap approach named limited overlap. The new approach gives to the user the means to specify over which vertices the overlap will be allowed. These vertices may represent significant elements in the context we work with. For example in the use-case presented in chapter 6.2, implementation of evacuation plans in case of volcanic contingences in the Popocatépetl volcano zone, vertices representing roads were allowed for overlap since these spatial objects represent relevant elements in the evacuation plans. Moreover, we visualized three motivation issues to propose the implementation of the new approach. First, we demonstrated by using limited overlap that we obtain a search space reduction in the substructure discovering process since we allow overlap but it is restricted to the elements stated by the user. Second, as result of a search space reduction we also get a processing time reduction. Remember as part of the substructure discovery process there exist a validation and discarding phase over the instances of a substructure. Instances no discarded may become candidates to further expansion in order to discover new substructures. Third, by giving the user the capability to state the set of vertices where overlap is allowed, we are orienting the search over particular overlapped instances and, at the same time, discarding no relevant overlapped instances.

In order to be able to demonstrate the feasibility, capacity to mine and to discover patterns by using a graph-based approach as proposed, we developed a prototype system implementing our model to create graph-based datasets, to mine those datasets (by calling the Subdue system), and to visualize the discovered patterns.

In chapter 6 we described three illustrative use-cases showing the applicability of our proposal. We used two test contexts from the real world: a population census from the year of 1777 in Puebla downtown, and a Popocatépetl volcano database. The generated results from those test contexts give us a panorama respect to how and to what we could achieve by using this approach. It is important to remark the fact that we can use this approach in any domain that can be represented as a graph.

We have shown the feasibility of our model for modeling and mining spatial data. In this context, perspectives related to enhance our overall work (graph-based model, data mining algorithm, and prototype system) include the following issues:

- **Visualization of discovered knowledge.** For example, visualization of discovered knowledge over the spatial layers, by using chart, and navigation through the discovered patterns hierarchy using a hypergraph approach.
- **Enhancing the algorithms used to create the graph-based datasets according to the proposed models.** Validation of spatial relations among spatial object is a phase that in most cases requires several computational resources. So, our algorithms must implement efficiently this task (i.e. discarding noise, using spatial indices, etc.).

- **Mining the graph.** We proposed and used the Subdue system as our data mining tool, moreover we implemented a new algorithm name limited overlap. Subgraph isomorphism is a NP-complete problem, so we must be able to face this problem in order that our processing times for discovering knowledge meet acceptable parameters of efficiency,
- **Relationships among non-spatial data describing spatial objects.** Implicit relations among attributes describing the spatial objects may be included in the model in order to enhance the representativeness of the data.

Spatial data mining is a promising research field. Several approaches have been developed, and without any doubt, new approaches will be proposed. It is a field in continuous improvement. Our contribution to the spatial data mining goes in that direction.

Chapter 8

BIBLIOGRAPHY

[1] AGRAWAL Rakesh, IMIELINSKI Tomasz, SWAMI Arun. Mining Association Rules between Sets of Items in Large Databases. **In** : Proc. of the 1993 International Conference on Management of Data, Washington D.C.. New York, NY : Association for Computing Machinery Press, 1993, pp. 207-216. ISBN 0-89791-592-5

[2] BERNHARDSEN Tor. Geographic Information Systems: An Introduction. 2nd ed. New York : John Wiley & Son Inc, 1999, 448 p. ISBN 0471419680

[3] CHEIN Michel, MUGNIER Marie-Laure. Conceptual Graphs: Fundamental Notions. Revue d'intelligence artificielle, 1992, vol. 6, n° 4, pp. 365-406.

[4] CHEN Ming-Syan, HAN Jiawei, YU Philip S. Data Mining: An Overview from a Database Perspective. Institute of Electrical and Electronics Engineers, Transactions on Knowledge and Data Engineering, 1996, vol. 8, n° 6, pp. 866-883. ISSN 1041-4347

[5] DIESTEL Reinhard. Graph Theory **[on line]**. 3rd ed. New York : Springer-Verlag, 2005. Available on :<http://www.math.uni-hamburg.de/home/diestel/books/graph.theory> (last visit 01.10.2005). ISBN 3-540-26182-6

[6] EGENHOFER Max J. A Model for Detailed Binary Topological Relationships. Geomatica, 1993, vol. 47, no 3-4, pp. 261-273.

[7] EGENHOFER Max J., FRANZOSA Robert D. On the Equivalence of Topological Relations. International Journal of Geographical Information Systems, 1995, vol. 9, no 2, pp. 133-152.

[8] EGENHOFER Max J., HERRING John R. Categorizing Binary Topological Relationships between Regions, Lines, and Points in Geographic Databases. Technical Report. Department of Surveying Engineering, University of Maine, Orono, 1991, 28 p.

[9] ESTER Martin, FROMMELT Alexander, KRIEGEL Hans-Peter, SANDER Jörg. Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support. Data Mining and Knowledge Discovery, 2000, vol. 4, no 2-3, pp. 193-216. ISSN 1384-5810

[10] ESTER Martin, FROMMELT Alexander, KRIEGEL Hans-Peter, SANDER Jörg. Algorithms for Characterization and Trend Detection in Spatial Databases. **In** : Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, New York, NY, 1998, pp. 44-50.

[11] ESTER Martin, KRIEGEL Hans-Peter, SANDER Jörg. Spatial Data Mining: A Database Approach. **In** : Proc. of the 5th International Symposium on Large Spatial Databases, Berlin, Germany, July 1997, pp. 47-66.

[12] FAYYAD Usama M., PIATETSKY-SHAPIRO Gregory, SMYTH Padhraic. Knowledge Discovery and Data Mining: Towards a Unifying Framework. **In** : SIMOUDIS Evangelos, HAN Jiawei, FAYYAD Usama M. Eds. Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, Oregon. Menlo Park, CA : American Association for Artificial Intelligence Press, 1996, pp. 82-88.

[13] FAYYAD Usama M., PIATETSKY-SHAPIRO Gregory, SMYTH Padhraic, UTHURUSAMY Ramasamy Eds. Advances in Knowledge Discovery and Data Mining. Menlo Park, CA : American Association for Artificial Intelligence/Massachusetts Institute of Technology Press, 1996, 625 p. ISBN 0-262-56097-6

[14] FAYYAD Usama M., SMYTH Padhraic. Image Database Exploration: Progress and Challenges. In : Proc. of the 1993 Workshop on Knowledge Discovery in Databases. Washington D.C., July 1993, pp. 14-27.

[15] FAYYAD Usama M., WEIR Nicholas, DJORGOVSKI S. SKICAT: A Machine Learning System for Automated Cataloging of Large Scale Sky Survey. **In** : UTGOFF Paul E. Eds. Proc. of the 10th International Conference on Machine Learning, 1993, University of Massachusetts. San Mateo, CA : Morgan Kaufmann, 1993, pp. 112-199.

[16] FRAWLEY William J., PIATETSKY-SHAPIRO Gregory, MATHEUS Christopher J. Knowledge Discovery in Databases: An Overview. **In** : PIATETSKY-SHAPIRO Greogry, FRAWLEY William J. Knowledge Discovery in Databases. Menlo Park CA : American Association for Artificial Intelligence/Massachusetts Institute of Technology Press, 1991, pp. 1-27.

[17] GONZALEZ Jesus A. Empirical and Theoretical Analysis of Relational Concept Learning Using a Graph-based Representation. PhD Thesis. Arlington : The University of Texas at Arlington, 2001, 138 p.

[18] GONZALEZ Jesus A., HOLDER Lawrence B., COOK Diane J. Structural Knowledge Discovery Used to Analyze Earthquake Activity. **In** : Proc. of the 13th Annual Florida Artificial Intelligence Research Symposium. American Association for Artificial Intelligence Press, 2000, pp. 86-90.

[19] GRAPHVIZ - AT&T RESEARCH. Graphviz - Graph Visualization Software **[on line]**. Available on : <http://www.research.att.com/sw/tools/graphviz/> (last visit 01.10.2005).

[20] HAN Jiawei, CAI Yandong, CERCONE Nick. Knowledge Discovery in Databases: An Attribute-Oriented Approach. **In** : YUAN Li Yan, Proc. of the 18th International Conference on Very Large Databases, Vancouver, British Columbia, Canada. San Mateo : Morgan Kaufmann Publishers, 1992, pp. 547-559.

[21] HAN Jiawei, FU Yongjian. Exploration of the Power of Attribute-Oriented Induction in Data Mining. **In** : [13].

[22] HAN Jiawei, KAMBER Micheline. Data Mining: Concepts and Techniques. San Francisco : Morgan Kaufmann, 2000, 550 p. ISBN 1-55860-489-8

[23] HOLDER Lawrence B., COOK Diane J., GONZALEZ Jesus A., JONYER I. Structural Pattern Recognition in Graphs. **In** : CHEN Dechang, CHENG Xiuzhen Eds. Pattern Recognition and String Matching. Dordrecht : Kluwer Academic Publishers, 2002, 772 p. ISBN 1-4020-0953-4

[24] HOLSHEIMER Marcel, KERSTEN Martin L. Architectural Support for Data Mining. CS-R9429. Amsterdam, The Netherlands : CWI (Centre for Mathematics and Computer Science), 1994.

[25] KAUFMAN Leonard, ROUSSEEUW Peter J. Finding Groups in Data: An Introduction to Cluster Analysis. New York : Wiley-Interscience, 1990, 368 p. ISBN 0471878766

[26] KNORR Edwin M., NG Raymond T. Finding Aggregate Proximity Relationships and Commonalities in Spatial Data Mining. Institute of Electrical and Electronics Engineers, Transactions on Knowledge and Data Engineering, 1996, vol. 8, n° 6, pp. 884-897.

[27] KOLATCH Erica. Clustering Algorithms for Spatial Databases: A Survey. University of Maryland, College Park : Department of Computer Science, 2001, 22 p.

[28] KOPERSKI Krzysztof, HAN Jiawei. Discovery of Spatial Association Rules in Geographic Information Databases. **In** : Proc. of the 4th International Symposium on Large Spatial Databases, Portland, Maine, August 1995, pp. 47-66.

[29] KOPERSKI Krzysztof, HAN Jiawei, ADHIKARY Junas. Spatial Data Mining: Progress and Challenges. **In** : Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery, Montreal, Canada, June 1996, pp. 55-70.

[30] KOPERSKI Krzysztof, HAN Jiawei, STEFANOVIC Nebojsa. An Efficient Two-Step Method for Classification of Spatial Data. **In** : Proc. of the 8th Symposium on Spatial Data Handling, Vancouver, Canada, 1998, pp. 45-54.

[31] LAURINI Robert. Information Systems for Urban Planning: A Hypermedia Cooperative Approach. London : Taylor and Francis, 2001, 368 p. ISBN 0748409645

[32] LAURINI Robert, MILLERET-RAFFORT F. Les bases de données en géomatique. Paris : HERMES, 1993, 330 p.

[33] LAURINI Robert, THOMPSON Derek. Fundamentals of Spatial Information Systems. London : Academic Press, 1992, 680 p. (The A.P.I.C. series, no 37) ISBN 0-12-438380-7

[34] LU Wei, HAN Jiawei, OOI Beng C. Discovery of General Knowledge in Large Spatial Databases. **In** : Proc. of the Far East Workshop on Geographic Information Systems, Singapore, June 1993, pp. 275-289.

[35] MUGNIER Marie-Laure. On Generalization/Specialization for Conceptual Graphs. Journal of Experimental and Theoretical Artificial Intelligence, 1995, vol. 7 pp. 325-344.

[36] NG Raymond T., HAN Jiawei. Efficient and Effective Clustering Methods for Spatial Data Mining. **In** : Proc. of the 20th Very Large Databases Conference, Santiago, Chile, 1994. San Francisco, CA, 1994, pp. 144-155.

[37] PECH PALACIO Manuel, SOL David, GONZALEZ Jesus A. Adaptation and Use of Spatial and non-Spatial Data Mining. Proc. of the International 2002 Workshop Semantic Processing of Spatial Data, Centre for Computing Research, Instituto Politécnico Nacional, México D.F., December 2002. ISBN 970-18-8521-X

[38] SHEIKHOLESLAMI Gholamhosein, CHATTERJEE Surojit, ZHANG Aidong. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. **In** : GUPTA Ashish, SHMUELI Oded, WIDOM Jennifer Eds. Proc. of the 24th International Conference on Very Large Databases, New York, NY, 1998. San Francisco, CA : Morgan Kaufmann Publishers, 1998, pp. 428-439. ISBN 1-55860-566-5

[39] SHEK Eddie C., MUNTZ Richard R., MESROBIAN Edmond, NG Kenneth. Scalable Exploratory Data Mining of Distributed Geoscientific Data. **In** : Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996. Menlo Park, CA : American Association for Artificial Intelligence Press, 1996, pages 32-37.

[40] SOL David, LOYO E., RAZO Antonio. Risk Management in the Popocatépetl Volcano Zone. **In** : Workshop on Advanced Techniques for the Assessment of Natural Hazards in Mountain Areas, Innsbruck, Austria, June 2000, pp. 97-101.

[41] STOLORZ Paul E, DEAN Christopher. Quakefinder: A Scalable Data Mining System for Detecting Earthquakes from Space. **In** : Proc. of the 2nd International Conference on Data Mining, Portland, Oregon, 1996. Menlo Park, CA : American Association for Artificial Intelligence Press, 1996, pp. 208-213.

[42] STOLORZ Paul E., NAKAMURA H., MESROBIAN Edmond, MUNTZ Richard R., SHEK Eddie C., SANTOS J. R., YI J., Ng K., CHIEN S. Y., MECHOSO C. R., FARRARA J. D. Fast Spatio-Temporal Data Mining of Large Geophysical Datasets. **In** : Proc. of the 1st International Conference on Data Mining, Montreal, Canada, 1995. Menlo Park, CA : American Association for Artificial Intelligence Press, 1995, pp. 300-305.

[43] SUBDUE SYSTEM - THE UNIVERSITY OF TEXAS AT ARLINGTON. SUBDUE - Graph Based Knowledge Discovery **[on line]**. Available on : <http://ailab.uta.edu/subdue> (last visit 01.10.2005).

[44] ZHANG Tian, RAMAKRISHNAN Raghu, LIVNY Miron. BIRCH: An Efficient Data Clustering Method for Very Large Databases. **In** : Proc. of the 1996 International Conference Management of Data, Montreal, Canada, 1996. New York, NY : Association for Computing Machinery Press, June 1996, pp. 103-114. ISSN 0163-5808