Dipartimento di Informatica e
Scienze dell'Informazione

Query Processing and Analysis of Multi-resolution Spatial Data in Distributed Architectures

by

Paola Podestà

Theses Series **DISI-TH-2010-03**

DISI, Università di Genova
v. Dodecaneso 35, 16146 Genova, Italy http://www.disi.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Query Processing and Analysis of Multi-resolution Spatial Data in Distributed Architectures

by

Paola Podestà

July, 2010

Dottorato di Ricerca in Scienze e Tecnologie dell'Informazione e della Comunicazione
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
http://www.disi.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Paola Podestà
DISI, Univ. di Genova
podesta@disi.unige.it

Date of submission: July 2010

Title: Query Processing and Analysis of Multi-resolution Spatial Data in Distributed Architectures

Advisors: Barbara Catania
DISI, Univ. degli Studi di Genova
catania@disi.unige.it

Alberto Belussi
Dipartimento di Informatica - Univ. degli Studi di Verona
alberto.belussi@univr.it

Ext. Reviewers: Eliseo Clementini
Dipartimento di Ingegneria Informatica ed Elettrica - Univ. degli Studi dell'Aquila
eliseo.clementini@univaq.it

Yannis Theodoridis
Department of Informatics - University of Piraeus
ytheod@unipi.gr

Abstract

In recent times, we assist to a fast increase of distributed architectures and environments (e.g., Web services, P2P networks, satellite networks, etc.) as platforms to access, share, and integrate any type of information. As a consequence, the availability of huge amounts of highly heterogeneous data has significantly grown up. Among all possible data that can be collected in such architectures, spatial data and, more precisely, geographical data, play nowadays an important role for an increasing number of applications and institutions. At the same time, the widespread availability of devices linked to location-aware technologies (LATs), such as global positioning systems, cell phones, in-vehicle navigation systems, wireless Internet clients, increases the importance of spatial data not only in decision processes of public and private institutions but also in everyday life.

Dealing with spatial data in distributed architectures requires to face with a complex environment where it is quite common to find multiple, i.e., multi-resolution, representations of the same or overlapping spatial datasets. Multi-resolution may have different meanings in different contexts. In the context of this dissertation, with multi-resolution we mean the assignment of different dimensions to the same geographical feature in distinct spatial datasets. From a system point of view, the presence of multi-resolutions datasets may lead to query processing and integration problems, since the same concept can be represented in different ways and at different resolutions. From a user point of view, this may result in a gap between the available data and the user's knowledge of such data during query specification, reducing user satisfaction in using a given application.

In order to cope with multi-resolution in distributed environments, several aspects have to be taken into account. Among them, the definition of multi-resolution aware query processing techniques is a relevant issue which may lead to an improvement of both query execution efficiency and user satisfaction, by increasing the quality of query processing results. Unfortunately, as far as we know, even if several approaches for modeling multi-resolution datasets have already been proposed, only few techniques have been provided exploiting multi-resolution information during query processing.

This PhD dissertation investigates how multi-resolution can be effectively exploited during query processing of spatial datasets. The basic idea is that of providing some mechanisms of query relaxation, by which the specific characteristics of multi-resolution datasets are taken into account and, as a consequence, approximated answers may be returned to the user. The techniques we propose are "qualitative", in the sense that they rely on the usage

of spatial relationships. Due to their importance in real applications, we restrict ourselves to consider topological and cardinal spatial relationships.

More precisely, as a first contribution, we present a formal model for topological and cardinal direction relationships in multi-resolution spatial datasets, suitable for defining relaxed query processing techniques. Then, we extend existing distance functions for both topological and cardinal relationships to deal with multi-resolution pairs of objects and we use them for consistency and similarity checking of multi-resolution spatial datasets.

As a second contribution, based on the provided similarity and consistency results, we propose a relaxed version of selection and join operators which, in case the query returns a small or empty set, relax the query condition in order to return an approximated result. To this purpose, different semantics are provided which introduce different types of relaxation in the query result, possibly satisfying different user needs. More precisely, under the Best Fit semantics, the smallest amount of relaxation is introduced in the query condition in order to return a non empty answer. On the other hand, under the Threshold semantics, the query is relaxed of a fixed amount, represented by a given system or user threshold. Finally, the Nearest Neighbor semantics, when the query cannot be executed on data at hand, relaxes the query with the more similar one, defined for the considered data.

In order to exploit the practical usability of the proposed operators, index-based processing algorithms, based on the usage of R-trees, are also provided. The proposed algorithms rely on a branch-and-bound approach and generate results in an ordered and progressive way, with respect to their similarity with the original query condition. Experimental results, performed over synthetic data, show that the overhead given by query relaxation is acceptable and that the proposed algorithms can be efficiently used in a real context.

To my family

Acknowledgements

Thank to my family and my friends that always have supported, helped, encouraged and trusted in me during all my studies. To Barbara Catania and Alberto Belussi the two reference professors of my PhD experience.

Then, a sincere thank to people that I met at DISI in Genoa during all these years...cited in a rigorous random order: Ivana, Anna, Sonia, Betta, Matteo, Marco, Anna, Daniela, Laura, Irene, Roberto, Valentina, Monica, Ilaria, Tatiana, Maria, Emanuele and all the people I forgot, the lunches in the garden, and in the common room, for the chats, the laughters and for the time they have spent with me during my life experience at DISI. A long chapter of my life is closed...and another is starting...

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This PhD thesis focuses on the development of techniques for the management of multi-resolution geo-spatial data in distributed and heterogeneous environments. Our purpose is to effectively exploit multi-resolution information in comparing geo-spatial datasets and during geo-spatial query processing.

This chapter is organized as follows. In Section 1.1, we informally introduce the issues concerning the management of multi-resolution geo-spatial datasets; in Section 1.2, we highlight the research problems related to the management of multi-resolution, addressed by this PhD thesis. Finally, the organization of this PhD thesis is described in Section 1.3.

1.1 Management of Multi-resolution Geo-Spatial Data

Geo-spatial data represent a strategic resource for public and private institutions, since the possibility of performing operations on such data, such as querying, comparing and analysing geo-spatial data, can improve decision processes and help experts in critical situations. Examples of geo-spatial data are objects coordinates in AUTOCAD systems, satellite images, geographic maps of a specific land or soil.

The issues concerning the modeling and the management of geo-spatial data have been investigated by the research community from about twenty years. At the beginning, the critical stage was the formal definition of geo-spatial concepts, models and operations in order to translate cartographic data into digital data and to automate cartographic processes. This task is not straightforward since geo-spatial data have a very complex structure. They are generally defined as data that refer to and describe objects in a multi-dimensional space. Geo-spatial data are associated with both *spatial* and *non-spatial attributes*. Spatial (geometric) attributes concern the specific position of an object in the

reference space, its specific extent, orientation, and dimension. On the other side, non-spatial attributes represent the non-geometric characteristics of data, such as the name of a nation, of a road, etc. Moreover, geo-spatial data are related by mutual geo-spatial relationships (i.e., topological, direction, and distance relations).

Spatial attributes (i.e, coordinates in a reference system) give a *quantitative* information of geo-spatial datasets, but they are not preserved under transformations such as scaling, translation or rotation. On the other side, geo-spatial relations are *qualitative* information that capture mutual relations between geo-spatial objects; additionally, some of them are invariant under transformations such as scaling, translation or rotation.

Geo-spatial data are usually represented by using two distinct models. Under the raster model, geo-spatial data are represented as the values of the cells (pixels) of a matrix (grid), where the cells can be pixels of aerial or satellite images or larger portions of territory collecting measures of physical phenomena (like atmospheric pressure, rainfall or soil type). Under the vector model, geo-spatial data are represented as points, curves, and regions, related through geo-spatial relations. The raster model does not directly represent geo-spatial relations among geo-spatial objects; on the other side, under the vector model both object geometric information (i.e, coordinates in a reference system) and topological relations between objects are explicitly represented. Both raster and vector model are used in Geographic Information Systems (GISs) for representing geo-spatial objects or phenomena. In this dissertation, we consider the vector model and we assume geo-spatial objects are organized in *maps*, representing sets of geo-spatial objects referring to different feature types (e.g., roads, buildings, rivers, etc.) on the reference space.

Nowdays, the growing availability of devices, applications, often distributed, and institutions collecting and managing geo-spatial data has radically modified management of data collections and in particular of geo-spatial data. Indeed, geo-spatial data collections on the net often satisfy the following properties: (i) they may refer the same geographical area but are highly heterogeneous because collected by different institutions; (ii) they are produced by different kinds of processes (e.g. social, ecological, economic), at different times; (iii) they need to be handled together (as in Geo-Spatial Data Infrastructure).

Data heterogeneity leads often to the availability of multiple and not coinciding representations of the same or overlapping geo-spatial maps, i.e., maps represented at different *resolution levels*. The term *multi-resolution* may have different meanings depending on the considered model. Under the raster model, resolution often represent cells (or pixels) size; thus, multi-resolution datasets correspond to multiple representations of the same map with different cell sizes. On the other side, under the vector model, multi-resolution is usually interpreted as the number of points used to represent a geometric value. Ad hoc operators have then been provided in order to generalize geo-spatial datasets objects in a controlled way and transform one map into a more general one through gradual changes. Another possible interpretation of multi-resolution under the vector model, quite relevant

for distributed environments, concerns the representation of the same object in distinct maps, using different geometric types. For example, a road can be represented as a region for an ecological process whereas it can be represented as a line for a traffic analysis process. In this dissertation, we consider multi-resolution maps according to this last meaning.

Managing multi-resolution datasets in a distributed environment is an interesting but complex problem, that can be addressed by considering different points of view. From a system point of view, multi-resolution may lead to query processing and integration problems, since the same concept can be represented in different ways and at different resolutions. On the other side, from a user point of view, multi-resolution may result in a gap between the available data and the user knowledge of such data during query specification, reducing user satisfaction in using a given application. To overcome such problems, multi-resolution should be exploited in geo-spatial querying and analysis processes. Examples of applications that may get benefits from the usage of multi-resolution aware techniques are:

1. Tools for consistency checking of maps obtained from different sources, in order to establish whether such maps may lead to contradictory results if a user performs some kind of analysis over them.

2. Tools for detecting and analyzing changes in time that occurred in a given area or across overlapping areas; these tools could be useful in order to identify the amount of differences existing between two maps before starting a more accurate comparison of their content.

3. Geo-spatial query processors, since, in presence of multi-resolution, the specification of equality-based queries, by which the user specifies in an exact way the constraints that data to be retrieved must satisfy, may not be the right choice.

4. Mediator systems, where it should be necessary to integrate distinct and possibly multi-resolution data sources in order to process global query requests or update operations.

Among the previous examples, two main issues involved in the management of multi-resolution geo-spatial datasets in distributed architectures can be identified:

- The first issue concerns the comparison of multi-resolution datasets, required for consistency checking and change analysis. To this purpose, we notice that geo-spatial relations are a good choice for checking map consistency. Even if both geometry and properties concerning geo-spatial relationships are available in a multi-resolution map, it seems reasonable to discard geometry. Indeed, geometric consistency would be reduced to an equality test between two geometric map representations and similarity will require a sort of object extension measure in order to compare the geometric changes between two objects; in both cases, after a change of resolution (i.e.,

dimension), these properties cannot be preserved. On the other hand, information about geo-spatial relationships is more abstract than the geometric one and describes properties that are preserved after object dimension changes.

- The second issue concerns query specification and processing. Among interesting queries, those based on geo-spatial, and more precisely topological relations are used in an increasing number of applications, including spatial and geographical applications, spatial reasoning, image and multimedia databases, and cognitive sciences. Unfortunately, the presence of multi-resolution datasets may generate a gap between the available data and the users knowledge of such data during query specification. Since multiple representations exist, the user may not exactly know the geo-spatial domain she wants to query, in terms of properties, available features, and geometric feature types. This gap may impact the quality of the results obtained by a query execution, in terms of completeness and accuracy, reducing user satisfaction in using a given application since the obtained result may not exactly correspond to user needs. In those cases, query relaxation can be used in order to provide a reasonable answer to the user even in absence of data which precisely satisfy the query conditions.

1.2 Research Problems and Thesis Goals

While several proposals exist for multi-resolution modeling and geo-spatial relations (see Chapter 2), only few approaches directly exploit geo-spatial relations and multi-resolution in query processing. In particular, as far as we know, no unified approach for assessing map consistency and similarity based on geo-spatial relationships in presence of multi-resolution data has been proposed yet. On the other hand, since in distributed environments geo-spatial datasets are often heterogenous and incomplete, data management techniques relying on 'qualitative' information like geo-spatial relations could be quite effective since information about geo-spatial relationships can often been extracted and used also in presence of approximated and incomplete data. Additionally, as far as we know, no approach has been proposed so far for relaxing queries based on topological predicates when they return an empty or insufficient answer, in order to improve result quality and user satisfaction.

Starting from those considerations, the first objective of this PhD thesis is to study and develop a framework for the evaluation of consistency and similarity between multi-resolution maps based on geo-spatial relations. Due to their importance in GIS applications, we mainly consider topological and direction relations. For them, we propose a model, obtained by extending other existing models, in order to cope with multi-resolution datasets. Then, we present some similarity functions, to be used for consistency and similarity checking of multi-resolution datasets, as well as for query relaxation. Based on the proposed

functions, we finally provide an overall framework for consistency checking based on qualitative information.

The second contribution is the definition of a general strategy for relaxing query operators. We focus in particular on topological selection and join queries and we provide different specific semantics addressing different relaxation issues. The *Best Fit semantics (BF)* returns geo-spatial objects which better satisfy the query condition. The *Threshold semantics (Thr)* relaxes the topological condition using a threshold value that represents the minimum accepted similarity of the topological query relation with respect to the relations existing between the considered features. The *Nearest Neighbor semantics (NN)* relaxes the query with the most similar ones defined for the considered objects. The proposed query algorithms rely on the usage of the R-tree index structure and are based on a branch and bound approach, which discards the visit of some R-tree sub-trees that cannot produce further results. The pruning condition is defined by using a topological similarity function and some compatibility rules, first proposed in [PTSE95] for regions, here extended to pairs of geo-spatial objects of arbitrary dimension. While the usage of a branch and bound approach for geo-spatial query processing is not new (see, e.g., [ZPZL05]), as far as we know, no other existing proposals rely on such an approach for executing relaxed geo-spatial queries. We believe that the proposed, non-trivial, extension of the branch and bound approach to the new context may also help in integrating the proposed techniques in existing systems.

As a third contribution, in order to show the efficacy and efficiency of the proposed relaxed query operators, this dissertation is completed with the presentation of a prototype system called *ARTjQA (Advanced Relaxed Topological java Query Analizer)*, developed in Java, implementing the designed query processing algorithms. A complete set of experimental results are also presented and discussed, showing that the overhead given by query relaxation is acceptable.

Thesis contributions can therefore be summarized as follows:

- Definition of a framework for checking map consistency and similarity through the analysis of topological and direction relations.
- Definition of some similarity functions to be used for consistency and similarity checking.
- Definition of a similarity function to be used for query relaxation.
- Definition of relaxed selection and join operators and of various semantics, addressing different relaxation issues.
- Implementation of a prototype for the experimentation of the presented relaxed geo-spatial operators.

- Extensive experimental activity, to quantify the overhead given by query relaxation.

1.3 Overview of the Thesis

The remainder of this PhD thesis is organized as follows.

Chapter 2 surveys existing proposals related to the thesis topics, concerning the management and the query processing of multi-resolution geo-spatial datasets.

Chapter 3 deals with consistency checking issues for multi-resolution datasets. More precisely, a model for topological and direction relations is first provided. Then, specific similarity functions are proposed, suitable for consistency checking and/or query relaxation. Based on the proposed functions, an overall consistency checking framework is then presented.

Chapter 4 presents relaxed selection and join operators, proposes various semantics, and introduces relaxed query processing algorithms.

Chapter 5 presents the developed prototype. Experimental results concerning selection and join operators, presented in Chapter 4, are also provided and widely discussed.

Finally, Chapter 6 concludes the thesis by summarizing the PhD research work contributions and by discussing several open issues and topics for future research.

One appendix is also provided, presenting an example for both configuration and data input files for the ARTjQA prototype.

Chapter 2

Related Work

In this chapter we present a survey of the existing theoretical works for the representation and query processing of geo-spatial data.

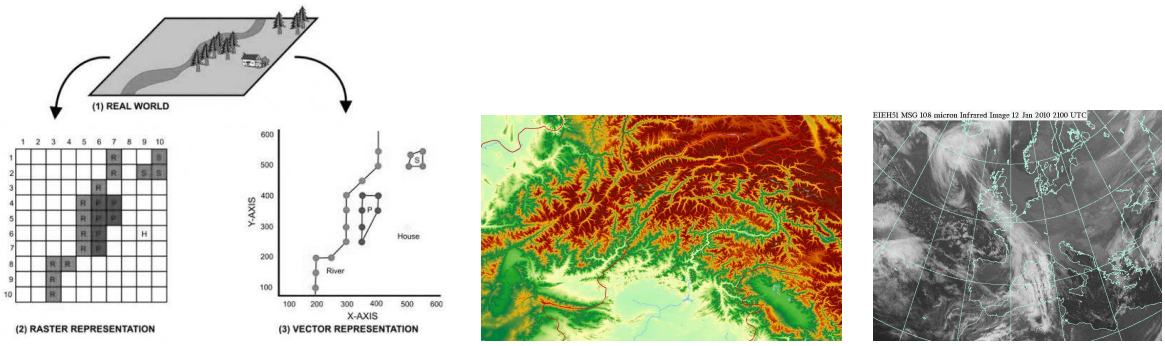
We start by describing some of the models proposed during the last two decades for the representation of both geo-spatial data (Section 2.1) and geo-spatial relations (Section 2.2). Then, we present some approaches for modeling multi-resolution datasets (Section 2.3). and for consistency and similarity checking (Section 2.4) of geo-spatial datasets.

Considering query processing, a survey of related work concerning both non-approximated and approximated techniques is presented in Section 4.5 and Section 2.6, respectively. Finally, in Section 2.7, we give an overview of the assumptions we take into account in this PhD thesis.

2.1 Geo-Spatial Data Models

The discussion around geo-spatial data have interested the research community in different areas including geometric modeling, computational geometry, Geographic Information Systems (GISs), from a very long time. In the early 1990th, as presented in [Fra92b, EH91, WD04], the main attempt was on the formalization of geo-spatial data models.

Geo-spatial data models are usually associated with two different perceptions of the world, and can be classified into *field models (or field views)* and *object models (or object views)* [Fra92b, Goo92]. In the first case, the space is represented as a surface (also called layer) partitioned into regular elements (e.g., grid of cells or pixels). The data in a layer represent a physical phenomenon or a characteristic of the geographic area considered that assumes



(a) (b) (c)

Figure 2.1: Geo-spatial models: (a) the same geo-spatial area represented under the raster and vector models; (b), (c) different examples of raster data.

a specific value in every element of the grid; the geo-spatial data represented on a layer usually do not overlap, thus they represent a partition of the considered space. Usually, a layer corresponds to a particular theme (type of information) such as the temperature of a city, the rainfall over a particular land, the classification of the utilization of a land, etc. Under this model, objects represented in the space are created during the modeling process and do not exist independently (e.g., the region with an average temperature in July of 30 degrees in a given city is not a real world object, but the result of a particular measurement).

On the other side, in the *object model* the real world is represented by a set of discrete geo-spatial objects, or entities embedded in the reference space. The fundamental abstractions used to represent geo-spatial objects are points, curves, and regions. A point is usually used for representing an object for which only its location in space, but not its extent, is relevant. For example, a lamppost or a mountain peak can be modeled as a point. A curve is the basic abstraction for networks, e.g., roads, rivers, cables for phone. A region is the abstraction for something having an extent in 2-dimensional space (e.g., a country, a lake, or a national park). A region may have holes and may also consist of several disjoint pieces. In the object model, the considered entities exist independent of their locations (e.g., a city, a lake, a river, etc.). Each object is described by its attributes, and can be identified using an unique attribute; two different objects can occupy the same position (i.e., the objects do not partition the reference space).

The *raster model* is an example of field model. It models and discretized the space as a set of regularly shaped and small sized areas, called cells, forming a regular tessellation. In practice, the reference space is divided using a grid; in this way, the space is discretized into finite little elements that have a specific geometry and shape, and also a specific value (see the grid on the left in Figure 2.1(a)). The coordinates in this case are not stored

but derived by the position of the cells. Square cells are called pixels. The single value attached to each cell is the result of the measurement of a natural phenomenon or the value obtained by a classification process, concerning a characteristic of the area involved. Any types of image regarding the earth surface can be considered a raster data (from areal photos to infrared satellite images, see Figure 2.1 (b) and (c)).

The *vector model* is an example of object model. The idea is to represent geo-spatial data by means of a set of objects having some thematic attributes and one spatial attribute, which is described by using a geometric value. Usually linear geometry is adopted and different geometric types are defined and used to specify the domain of spatial attributes. There are three fundamental geometric types: point, curve and surface, with the corresponding linear implementations: point, polyline and polygon. Each geometry is usually represented using coordinate pairs in a given reference system. In particular in the two dimensional space, each point is represented by a pair of coordinates; each polyline, by a sequence of points and each polygon by a set of closed polylines.

Several specification of geo-spatial types, under the vector model, have been proposed. In particular, the one proposed by the Open GeoSpatial Consortium (OGC), called Simple Feature Geometry [OGCc], has been implemented in many systems. In such a model, any geo-spatial object is called *feature*. Each feature has both spatial and non-spatial information. Basic types for spatial information are points, curves, and surfaces. A Point is a 0-dimensional geometry, that represents a single position in the coordinate space. The boundary of a point is empty. A Curve is a 1-dimensional geometry, stored as a sequence of points; type LineString specializes Curve when linear interpolation between such sequence of points is used. A curve is simple if it does not intersect itself; it is closed when the two end points are the same, thus its boundary is empty; otherwise, the boundary of an open curve is composed of the two end points. A Line is a LineString with exactly two end points. The Surface type is a 2-dimensional object possibly with holes. The boundary is the set of both external and internal closed curves (representing holes) that correspond to its boundary. A Polygon is a planar surface possibly with holes.

The development of different geo-spatial vector and raster models has been deeply studied in the last four decades (see [GB90, MP94, Par95, EGG$^+$99, SCR$^+$99, GYC07, G̈94, WD04, RHE$^+$04]); both models are used in GIS to represent geo-spatial data, but since in this thesis we are interested in the management of sets of geo-spatial objects, in the following, we focus our attention on the vector model.

Several methods have been proposed so far for the representation of geometries under the vector model. Such methods are based on different mathematical theories: in particular, Euclidean geometry and mathematical topology (see [Ale61]), which, in our specific case, is applied to point-sets and is called point-set topology. Mathematical topology studies the properties of topological spaces; in particular, the notions of open and closed sets; interior and closure; neighborhood and closeness. It also analyses and describes spatial properties

that are preserved under continuous deformations of objects (for example, deformations that involve stretching, but no tearing or gluing), called topological properties. These properties are important in GIS context, since very often, dealing with geo-spatial data, we need to preserve the topological properties of geometries we are manipulating, while by using pure Euclidean geometry for objects representation many problems can occur in processing and querying such data. For example, using Euclidean geometry a point in the plane is represented by a pair of coordinates, which are real numbers. Unfortunately, in computer architecture real numbers can be stored only by using approximated methods, thus only a discrete grid of points can be represented on the computer. Now, if we compute the intersection between two segments, it can produce a point that must be rounded to the nearest grid (i.e., representable) point and, as a consequence, it will not be a point of the intersecting segments.

Different approaches have been proposed in order to face these kind of problems. We present the two main methods for representing vector data, which have been implemented also in real GIS systems.

1. *Geometries as independent values*: the system stores the geometry of all database objects independently. This method is based on the representation of each geometry as an independent sequence of points of the Euclidean plane. The main drawbacks of such models is that it is not capable of preserving topological properties between objects under transformation such as rotation or scaling (see [G̈88]); moreover, some operations, like check for topological relations, need expensive algorithms (e.g., the calculation of intersections). On the other side, as independent values, geometries can be easily updated or inserted. Also visualization operations are not very expensive, they only need to access the geometries of objects, which is directly available.

2. *Geometries built on top of a common topological structure*: this approach is based on the separation between coordinates of points and objects with their geo-spatial relations (e.i., topological, direction and metric) in two different levels. Considering in particular the topological relations, two approaches have been proposed in literature. One approach is based on the simplices, that represent the elementary geometry in a given dimension. In particular, a 0-simplex represents a point, a 1-simplex represents a segment, and a 2-simplex represents a triangle, a 3-simplex a tetrahedron. Simplices constitute the building blocks for more complex geometries, called simplicial complexes. In the common structure, a set of simplices are stored together with their topological relations: in particular, the incidence relation (e.g., incidence of two points or of a point that lies on a segment) and the inclusion relation (e.g., inclusion of one point in a segment). In practice, the geometries are represented using two layers: the geometric layer where all geometries are represented in an abstract way by a set of simplices; (ii) the topological layer where topological relations between the simplices are explicitly stored. The separation between topology from

geometry is a critical issue in GISs, since it allows the system to maintain topological consistency among objects to be represented discarding the problems deriving from numerical approximation. Such approach is presented in [FK86, EFJ90]; then, several other models based on cell complexes (a generalization of simplicial complexes) have been developed, considering not only two-dimensional but also n-dimensional cell complexes (see [DMP93, PD95, Ber98, Pig94] to cite a fews).

An alternative proposal is based on the concept of a realm. Also in this case the basic idea is to propose a model based on a discrete geometry. In particular, a realm is defined as a finite set of points and line segments over a discrete grid of points satisfying these properties: (i) each point or end point of a line segment is a grid point, (ii) each end point of a line segment is also a point of the realm, (iii) no realm point lies within a line segment (i.e., on the line segment without being an end point) and (iv) no two realm segments intersect except at their end points. Also in this case the idea is to form the geometries of geo-spatial objects by composing their geometry starting from realm points and segments (see ([GS95])).

Usually in Geographical Information Systems (GISs), geo-spatial data, especially two-dimensional data represented according to the vector model, are grouped together into distinct datasets called *geographic plane maps* or simply *maps*. This approach is useful for the following reasons: a) geo-spatial data can be produced by different survey processes and may have different characteristics in metric precision, number of details and content type; b) distinct geo-spatial datasets can describe information of different nature that very rarely are manipulated together (for example, the map of the county borders inside a region and the map containing the soil taxonomy of the same region); c) often, geo-spatial data related to a specific geographic area is represented in distinct datasets, at different resolution levels. A map represents both spatial and non-spatial information of a set of geo-spatial objects, as well as geo-spatial relations among them.

2.2 Models for Geo-Spatial Relationships

Geo-spatial relations are binary relations that relate two geo-spatial objects, specifying their mutual position in the reference space. Geo-spatial relations can be classified into topological, direction, and metric relations. Topological relations consider connectivity and adjacency of objects, metrical relations quantify the distance between objects, using a distance function, and direction relations describe how objects are placed relative to each others.

In the following, we present some models for the representation of topological (Section 2.2.1), direction (Section 2.2.2) and metric (Section 2.2.3) relations.

2.2.1 Topological Relationships

Topological relations capture the essential geo-spatial relationships between geo-spatial objects; they consider connectivity and adjacency of objects and are invariant under continuous transformations of space such as translation, rotation, and scaling.

Due to the importance of topological relations for the management of geo-spatial data, several formal and non-formal definitions for them have been provided. First attempts to formalize topological relations were limited to one-dimensional data [All83]. Afterwards, the proposed models for topological relations considered two-dimensional data and were based on simplicial complexes, that describe a geo-spatial object in terms of a collections of primitive (points, segments, polygons) and disjoint objects, whose union is the object itself [Ege89, FK86]. A more general approach, based on point-set topology [Mun66], defines topological relations in terms of set operations using pure set theory; objects are point-sets and topological relations between objects are defined considering the set operations $=$, $\neq$, $\subseteq$, $\cup$. However, this approach does not lead to the definition of a complete and mutual exclusive set of topological relations. To overcome such problem, two extensions have been provided, one representing a geo-spatial object with two different point-sets: interior and boundary, and another considering three different point-sets: interior, boundary, and exterior. The boundary is the border separating the object from the rest of the space; the interior is the core object excluding the boundary; the exterior, the space outside the whole object (interior and boundary). Such approaches, called respectively *4-intersection model* and *9-intersection model*, are widely adopted as reference models for representing topological relations among regions, lines and points in vector maps. For this reason, in the following we describe them in details.

Under the *4-intersection model* [EF91], geo-spatial objects are modeled as point-sets describing their interiors and their boundaries, as shown in Figure 2.2(a). Only regions are considered and topological relations are defined as 2×2 matrices, containing the value empty ($\oslash$ or 0) or non-empty ($\neg\oslash$ or 1) for each intersection of the point-sets of the objects involved (see Figure 2.2(b)). As an example, Figure 2.2(c) presents the 4-intersection model configuration for the topological relation `overlap` between two regions. The 4-intersection model has subsequently been extended to consider also object exterior [EF95], leading to the definition of the *9-intersection model*. This model defines a formal categorization of binary topological relations between regions, lines, and points. In the 9-intersection model, each geo-spatial object A is represented by 3 point-sets: its interior A°, its exterior A^-, and its boundary ∂A (Figure 2.2(d)).

The definition of binary topological relations between two geo-spatial objects A and B is then based on the 9 intersections between each pair of object components. Thus, a topological relation can be represented as a 3×3 matrix, called *9-intersection matrix* (Figure 2.2(e)). As an example, Figure 2.2(f) presents the 9-intersection model configuration for

the topological relation `overlap` between two regions. By considering the value empty or non-empty for each 9-intersection, one can categorize in a complete way all the relationships between regions, lines, and points embedded in $\mathbb{R}^2$ [EH90].

The 9-intersection model have two main drawbacks: it is not able to distinguish between similar but different cases, such as two areas that have in common a point of their boundaries and two areas that have in common a line of their boundaries. The other drawback is that the 9-intersection model defines a too large number of different relationships, each corresponding to a particular configuration of the 9-intersection matrix, that are not very easy to manage for an user.

In order to overcome such drawbacks, in [CDFVO93], the 9-intersection model has been extended by considering the dimension of the intersection, that is, the value empty/non-empty of each intersection is replaced by the dimension of the intersection itself (e.i., empty, 0D, 1D and 2D), in order to further distinguish topological relations. To reduce the number of topological relations, the authors further provide a categorization of such relationships into only five intuitive and mutually exclusive topological relations (`disjoint, touch, in, cross, overlap`), shown in Figure 2.2(g)-(m) for regions. Such relationships are now at the basis of query processors of well known geo-spatial systems, such as Oracle 10g [ora] and ArcGIS ESRI [arc], of spatial Java APIs, such as JTS Topology Suite [JTS], and also adopted as reference model in the OpenGIS Simple Features [OGCc].

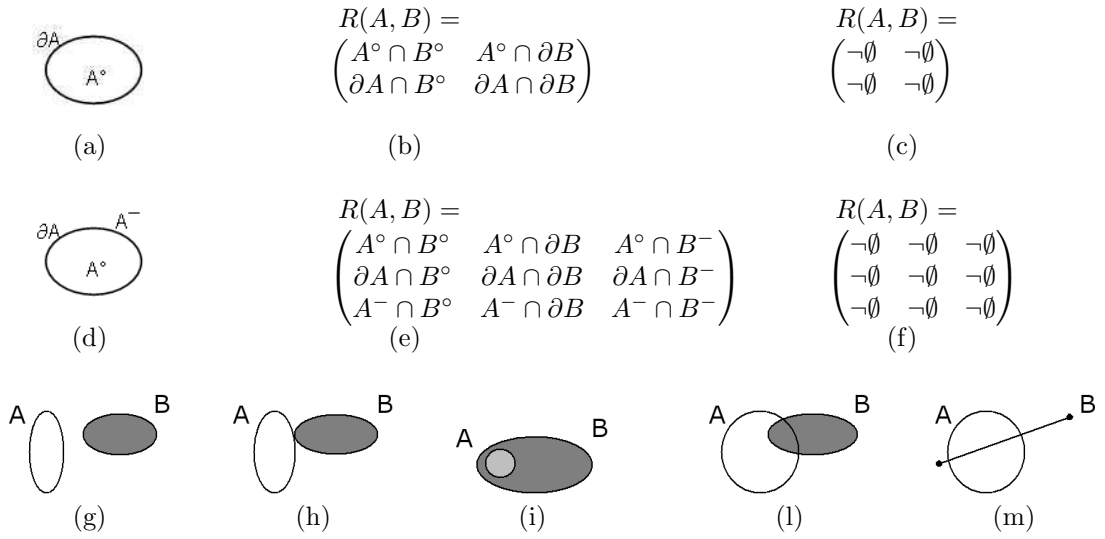
It is important to remark that not all relationships of the reference model can be defined for any pair of dimensions; for example relation *overlap* (see Figure 2.2(m)) is defined only between pairs of regions or pairs of lines.

The model defined in [CDFVO93] has been extended in [CDF96] to cope with complex features, that are complex area features (areas made up of several components possibly with holes), complex line features (lines with separation, self intersection, and an arbitrary number of end points) and complex point features (set of points considered as a unique entity).

2.2.2 Direction Relationships

Direction relations provide a way to determine what is the relative position of a target object with respect to a reference object, by considering some fixed directions.

The idea behind the representation of direction relations in [Fra92a, Fra96] is to map quantitative information (i.e., the degrees of an angle) into a set of symbols. More precisely, direction relationships are binary functions that map two object $(O1, O2)$ in the plane (representing, respectively, the reference object used to define directions and the target object whose direction with respect to the reference object has to be detected) onto a



$$R(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B \\ \partial A \cap B^\circ & \partial A \cap \partial B \end{pmatrix} \qquad R(A,B) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

$$R(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \qquad R(A,B) = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$$

Figure 2.2: Models for topological relationships: (a) object representation in the 4-intersection model; (b) the 4-intersection matrix; (c) 4-intersection matrix for `overlap` between regions; (d) object representation in the 9-intersection model; (e) the 9-intersection matrix; (f) 9-intersection matrix for `overlap` between regions; (g) A `disjoint` B; (h) A `touch` B; (i) A `in` B; (l) A `overlap` B; (m) B `cross` A.

symbolic direction d.

The number of direction symbols available depends on the model of directions used. The basic models for representing directions divide the space around the reference object in cone-shaped (or triangular) areas or in half planes, as shown in Figure 2.3(a)-(c). When the reference object is not a point, the most used model, called D_9 model, is based on the space decomposition presented in Figure 2.3(d) and approximates the reference object with its minimum bounding box (*MBB*). The space around the reference object is divided into distinct areas, called *tiles*, which are defined by extending to infinite the sides of the reference MBB. The resulting set of tiles are the usual set of directions {`N, NE, E, MBB, SE, S, SW, W, NW`}. We notice that all tiles, excluding *MBB*, are unbounded. Their union coincides with $\mathbb{R}^2$.

Based on the D_9 model, direction relations between two regions can be represented as 3×3 matrices containing the value empty or non-empty for each intersection between the target object A and the tiles generated by a reference object B (see Figure 2.4(a)). In [GE01, Goy00], this basic model has been extended to deal with regions without holes, lines, and points. In this case, a direction relation is represented again using a 3×3 matrix, but now each cell contains a 9 cells vector, called *neighbor code* (Figure 2.4(b)). A neighbor code

records information concerning the intersections between the boundaries of a particular tile and the target object A, using nine bits (x_0-x_8). Bit 0 (x_0) records the value of the intersection with the direction tile the vector refers to, say DT, and bits x_1-x_8 record the values of the intersections with the left (L), bottom-left (BL), bottom (B), bottom-right (BR), right (R), top-right (TR), top (T), and top-left (TL) boundaries of T, respectively. Each neighbor code corresponds to a binary number between 0 and 256. Thus, each matrix can be seen as a 3×3 matrix of integer numbers. Different matrix configurations correspond to different direction relations. However, by considering only connected objects, not all possible configurations represent a correct direction relations. The information contained in neighbor codes can also be represented by using a 5×5 matrix, called *direction matrix* (see Figure 2.4(c)). In such a matrix, a row and a column exist for each tile interior and each boundary between two tiles, according to the space subdivision presented in Figure 2.4(d). Each matrix element can assume the value empty or non-empty, depending on whether the target object A intersects or does not intersect the corresponding portion of space.

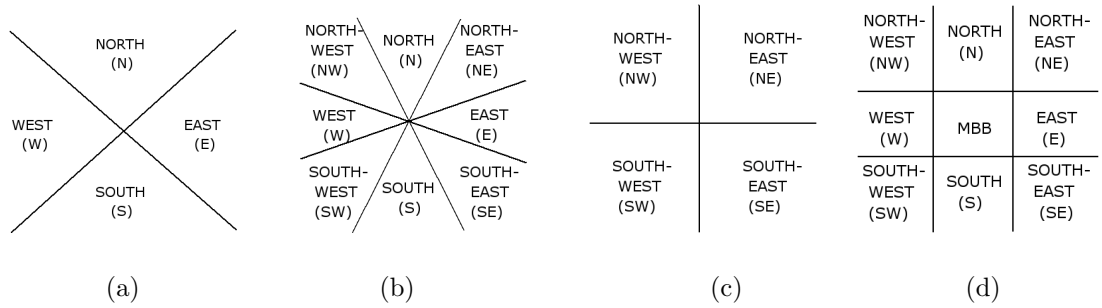
A formal model for direction relations for connected and disconnected regions, lines, and points is provided in [SK04, SK02], based on the 5×5 matrix.

In particular, the cells of the matrix represent the following elements (see Figure 2.4 (d)):

- the 9 two-dimensional areas representing the main directions. These areas correspond to the bounding box and the 8 cardinal directions, that is, the `MBB` (i.e., the minimum bounding rectangle of the reference object), `S, SW, W, NW, N, NE, E, SE` . Notice that each area does not include the parts of the axis forming it;
- the 8 semi-lines formed by the vertical and horizontal lines that start from the corners of the bounding box of the reference object. These semi-lines are denoted by `LSW, LWS, LWN, LNW, LNE, LEN, LES, LSE`. Each semi-line does not include the corner of the bounding box;
- the 4 line segments corresponding to the sides of the minimum bounding box of the reference object, denoted by `LS, LW, LN, LE`. Notice that each line segment does not include the corners of the bounding box;
- the 4 points representing the corners of the minimum bounding box of the reference object, denoted by `PSW, PNW, PNE, PSE`.

2.2.3 Metric Relationships

Metric relations capture distance information between objects in the reference space. Distances can be represented by using a qualitative or a quantitative approach. Under the



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

Figure 2.3: Different space divisions: (a) four cones; (b) eight cones; (c) four half planes; (d) nine-tiles.

quantitative approach, a distance function maps each pair of objects into a positive number, for example using Euclidean distance value. On the other side, under the qualitative approach, distances are represented as symbolic values. The qualitative approach for the representation of distance relations has been initially presented in [Fra92a] and further extended in [HCDF95]. It is based on the concept of *distance system*. A distance system consists of a list of distance symbols and a set of relations, usually $=$, $\neq$, $<$, $>$, $\leq$, $\geq$, that describe how the distance relations are mutually related. Three elements are needed to define a distance relation: the primary object, the reference object, and the frame of reference, representing the context in which the distance is determined. Qualitative distance values ideally partition the physical plane into circular regions of different sizes centered around a reference object (see Figure 2.5 as an example). Different levels of granularity are possible based on the number of different qualitative distance values considered.

2.3 Models For Multi-resolutions Geo-spatial Maps

The management of geo-spatial data at different levels of resolution is an important issue in the context of GISs. As already pointed out in Section 2.1, the term *multi-resolution* may have different meanings depending on the considered context. Under the raster model, resolution corresponds to the size of the cells (or pixels) of the grid used to represent the considered geo-spatial dataset. Thus, multi-resolution datasets correspond to multiple representations of the same geo-spatial dataset using different grid cell sizes. More detailed and precise representations of the space and, as consequence, smaller cell sizes, leads to higher resolution levels. On the other side, coarser representations of the space and larger cell sizes corresponds to lower resolution levels (see Figure 2.6).

Under the vector model, multi-resolution can be interpreted under two main different ways: (i) through the definition of ad hoc operators to generalize/specialize geo-spatial datasets

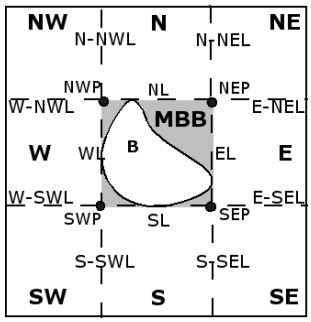
$$dir_{RR}(A,B) = \begin{pmatrix} NW_B \cap A & N_B \cap A & NE_B \cap A \\ W_B \cap A & MBB_B \cap A & E_B \cap A \\ SW_B \cap A & S_B \cap A & SE_B \cap A \end{pmatrix}$$

(a)

(b)

$$D(A,B) = \begin{pmatrix} NW_B \cap A & N\text{-}NWL_B \cap A & N_B \cap A & N\text{-}NEL_B \cap A & NE_B \cap A \\ W\text{-}NWL_B \cap A & NWP_B \cap A & NL_B \cap A & NEP_B \cap A & E\text{-}NEL_B \\ W_B \cap A & W_B & MBB_B \cap A & E_B \cap A & E_B \cap A \\ W\text{-}SWL_B \cap A & SWP_B \cap A & SL_B \cap A & SEP_B \cap A & E\text{-}SEL_B \cap A \\ SW_B \cap A & S\text{-}SWL_B \cap A & S_B \cap A & S\text{-}SEL_B \cap A & SE_B \cap A \end{pmatrix}$$

(c)



(d)

Figure 2.4: Models for direction relationships: (a) the 3×3 direction matrix; (b) the neighbor code; (c) the 5×5 direction matrix; (d) space subdivision corresponding to the direction matrix.

objects in a controlled way in order to transform one map into another through gradual changes, reducing the complexity of geometric values; (ii) through the representation of the same object in different datasets, using different geometric types.

Generalization approaches of the first type have been proposed in [ECDF94, PD95, Ber98], to cite a few. In [ECDF94], only regions are considered and multi-resolution is modeled as a set of different levels S_0,...,S_n, where each level is a set of geo-spatial objects which share some topological relations. Level $S_{(i+1)}$ is derived from level $S_{(i)}$ by using some generalization operations. The proposed operations guarantee to transform a map into a more general map which is similar to the previous with respect two criteria: object-similarity and relation-similarity. Two levels are object-similar if the topology of the common objects is unchanged, e.g., their dimensions are the same and there exists a continuous deformation that maps one object into the other one, even if their position and their mutual topological relations may change. On the other side, two levels are relation-similar if the mutual topological relations between objects do not change. For example, by considering Figure 2.6(e), map M2 is object-similar to M1, since the topology of the common objects is unchanged. On the other hand, since the position of PA2 as well as the topological relations between PA2, PA1 and R5 are changed, M2 is not relation-similar to M1. Further, M3 is relation-similar to M1, since mutual topological relations do not change. However,

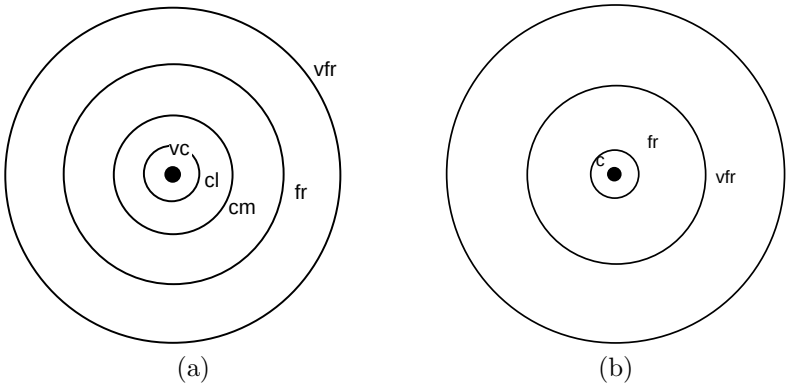


Figure 2.5: Examples of different distance systems: vc = very close; cl = close; cm =medium close; fr = far; vfr = very far.

it is not object-similar to M1, since the topology of object PA2 changes.

In [PD95, Ber98], points, lines, and regions are considered and geo-spatial maps are represented as abstract cell complexes. Intuitively, an abstract cell complex represent geo-spatial objects by decomposing them into disjoint elementary entities, related by topological relations.

In [PD95], multi-resolution is defined as a sequence of map simplification operations, which guarantee topological and metric consistency between the obtained maps. This model has been further formalized and extended in [Ber98] through the definition of a minimal and sufficient set of atomic generalization operators able to build intrinsically consistent multiple representations of maps at different level of details. Figures 2.6(b), (c), and (d) present some examples of atomic operators: (b) contraction of an open line to a point; (c) contraction of a region to a point; (d) contraction of a region to a line. This approach has been applied in [BE01] to the problem of progressive transmission of vector maps data over the Web. The basic idea is to pre-compute and store a sequence of consistent representations at different levels of details on the server, that are then transmitted to clients with increasing details.

Under the second interpretation, multi-resolution modeling is based on the concept of *feature type (ft)* [OGCc], that represents a class of real geo-spatial objects (e.g., rivers, roads, towns, etc...) (see [BBB+04]). In this thesis, we call this model *feature type-based map model*. Each feature type has some descriptive attributes and a spatial attribute, having a given dimension. Feature type instances are called *features*. According to OGC (Open Geospatial Consortium) [OGCa], any geo-spatial object can be of type: (i) point (dimension 0), (ii) curve - also called line - (dimension 1) or (iii) planar surface - more generally called region - (dimension 2).

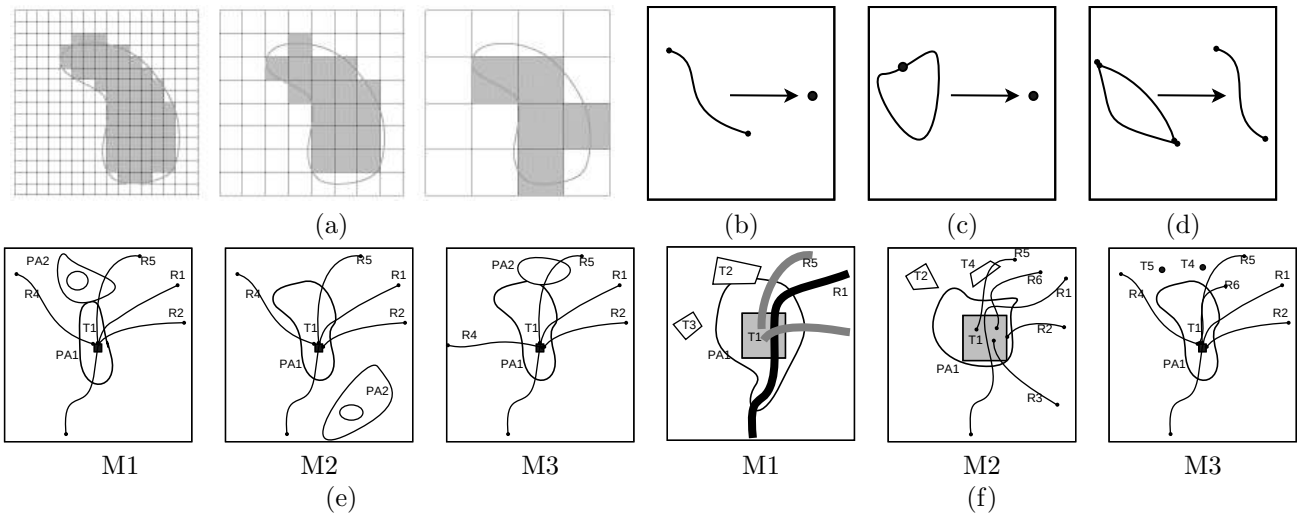


Figure 2.6: Examples of multi-resolution data: (a) different resolution levels in the raster model; (b),(c),(d) examples of atomic generalization operators; (e) map examples: M1is object similar to M2 and relation-similar to M3; (f) example of multi-resolution maps in the feature type model.

A *map schema* MS is a set of feature types. Given a set of map schemas $MS_1, ..., MS_n$ and a set of feature types $ft_1, ..., ft_m$, a feature type may belong to one or more map schemas, possibly with different dimensions. Thus, each feature type is assigned multiple dimensions, one for each map schema in which it appears.

Each instance of a map schema is called *map* and corresponds to a set of features, instances of the feature types belonging to the map schema. Given a set of map instances $M_1, ..., M_n$, instances of the map schemas $MS_1, ..., MS_n$, respectively, a feature f_j over a feature type ft_i may belong to one or more maps, associated with possibly different geometries and dimensions according to the map schemas. When this happens, we say that $M_1, ..., M_n$ are *multi-resolution maps*.

Example 1 *Consider the maps in Figure 2.6(f). They contain instances of three feature types: Road (features: R1, R2, R3, R4, R5, R6); Towns (features: T1, T2, T3, T4, T5; Pollution area (features: PA1). The maps are multi-resolution representations of the same datasets, according to the feature type-based map model. Indeed: feature T1 has dimension 2 in M1 and M2 and dimension 1 in M3; features R5 and R1 has dimension 2 in M1 and 1 in both M2 and M3.* □

2.4 Consistency and Similarity of Geo-Spatial Maps

Managing multi-resolution geo-spatial datasets is a very complex activity. One of the most critical issue is to establish whether such multiple representations may lead to contradictory information. To this purpose the concepts of consistency and similarity may be usefully exploited.

Informally, *consistency* describes the absence of any logical contradiction within a model of reality. Two different representations of the same geo-spatial dataset are inconsistent if they violate certain consistency constraints.

Example 2 *Consider objects $B1$, $B2$ in Figure 2.7 representing two buildings, modeled as regions in map M and as points in map M'. The buildings are disjoint in M but coincide, thus they are equal, in M'. Intuitively, it is obvious that maps M and M' are not topologically consistent. Formally, this can be proved since no atomic operators proposed in [Ber98], when applied on map M can return map M'.* □

On the other side, the concept of *similarity* is strictly related to the concept of *equivalence*. Generally speaking, two different geo-spatial datasets containing different representations of the same geo-spatial objects are equivalent if both geo-spatial relations among objects and the general structure of objects are preserved. In real situations, equivalence is a too strict condition to be satisfied; it seems more useful and interesting to consider the concept of similarity, that can be considered as a measure of deviation from equivalence.

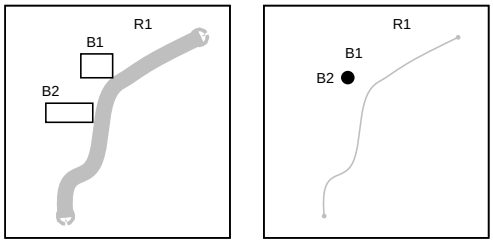
Consistency and similarity can be defined by considering various map properties. Choosing a property corresponds to focus on a specific abstract model for multi-resolution maps and to check whether the information they contain is contradictory or not with respect to that property.

In the following, we present the existing approaches for modeling and checking consistency and similarity of geo-spatial dataset taking into account topological relations (Section 2.4.1) and direction relations in (Section 2.4.2).

2.4.1 Topological Consistency and Similarity

Due to the importance of topological relations in the geo-spatial domain, the ability to evaluate consistency and similarity of different representations of topological relations is a critical task.

Several different approaches have been proposed, to face the issues related to consistency of multiple representations of spatial information. One approach focuses on the consistency



(a) Map M (b) Map M'

Figure 2.7: Example of non-consistent maps.

of existing multiple representations of geo-spatial objects, without considering how such representations have been obtained. The method is based on the 4-intersection model (Section 2.2.1) and considers the changes of the topology of the considered objects and of the topological relations among them [ECDF94]. Another approach considering complex geo-spatial objects structured as a network, a data structure commonly used for geo-spatial data (e.g., the roads of a nation, the electrical, telephone cable of a city, ect...), is presented in [TE96]. In this case, the focus is on the assessment of consistency among multiple representations of a network or of a network and another geo-spatial object, in particular a region or a line. Such model is based on an extension of the 9-intersection model (Section 2.2.1). Also the generation of consistent representations of a particular geo-spatial object is an important issue in geo-spatial context. An approach to this problem, based on the 4-intersection model, is presented in [TE97] concerning the assessment of consistency among multiple representations of geo-spatial objects obtained from the application of the aggregation operation. This operation is used to create multiple representations of the same geo-spatial dataset (see Section 2.3).

Concerning similarity, the approach presented in [EF95] defines a method for assessing similarity between two topological relations between regions represented using the 4-intersection model taking into account the content of the two matrices and other metrics. A framework defining a method for checking similarity between topological relations defined upon regions, defined according to the 9-intersection model, is presented in [EAT92]. Similarity is computed by comparing two 9 intersection matrices and computing the number of different values of each 9-intersection (topology distance). Using this function, a partial order over topological relations, called conceptual neighbor, (a concept introduced in [Fre91]) between regions can be defined and used to evaluate how two relations are far from each other. Topology distance is also used in [BE96] to evaluate similarity of geo-spatial scenes, by taking into account also direction and distance relations. and in [EM95] to define a model (snapshot model) to compare two different topological relations between lines and regions. In all the approaches cited above, similarity is computed only between pairs of features with the same dimension. Multiple feature representations are

not considered at all.

In the following we present all the introduced approaches in more details.

The framework presented in [ECDF94] considers a set of geo-spatial objects S related through topological relations and a sequence of different representations of S called levels: S_0,..,S_n. Level $S_{(i+1)}$ is derived from level $S_{(i)}$ by using some generalization operations. In the assessment of consistency between two consecutive levels the changes considered are those in the topology of objects and those in the topological relations among them. Changes in object dimensions are not considered. Based on such changes different types of conditions for assessing consistency are presented. The strongest condition requires that two consecutive leves have an a homeomorphic configuration, that is both objects topology and topological relations among objects do not change. Different degrees of deviations from an homeomorphic configuration are evaluated using two less strict criteria. One is the object-similarity, which assesses if the objects in two subsequent levels of representation are topologically similar, essentially, they do not change their dimension and topology, while topological relations could change. The other is the relation-similarity, which assesses if the topological relations in two subsequent levels of representation do not change, even if the topology of the objects change. The assessment of both object- and relation- similarity is based on some properties of the boundary-boundary intersection of the 4 intersection matrix.

A different approach addresses the problem of checking consistency between different representations obtained from the application of the aggregation operations is presented in [TE97]. In this approach regions with disconnected parts are considered and the focus is on the assessment of the consistency when merging several parts of an object into a whole. During this operation the critical aspect considered is that the geo-spatial relations with respect the other geo-spatial objects must be preserved. The approach presented provide a formalism to compare the situations before and after the aggregation operation and to assess whether they are consistent or not. The approach, based on the 4-intersection model, considers regions with disconnected parts and the basic idea presented is to derive from the relation among the parts the set of possible consistent relations with the aggregate. Also in this case changes in object dimensions are not considered and no similarity measures are provided. In [TE96], consistency among networks, defined as sets of lines (homogeneous networks) and sets of lines and regions (heterogeneous networks), is investigated, by providing several sufficient conditions. In particular, the consistency between scenes representing both homogeneous and heterogeneous networks at different levels of detail is considered when small changes occur from the one level to the next, such as the change of a segment length, or the change of the angle between two segments. The assessment of the consistency among networks (homogeneous and heterogeneous) is based on the concept of topological equivalence of networks. Informally, two homogeneous networks are topologically equivalent whether all segments in one network have corresponding segments in the

other network, and whether the topological relations, represented with the 9-intersection model, among the segments are the same in both networks. On the other side, for heterogeneous networks the only difference is that it is necessary to consider also regions an their corresponding relations.

For the assessment of the similarity of topological relations between connected regions a method based on the 4-intersection model is presented in [EF95]. Since comparing only the content of two 4 intersection matrices it is possible that two different configuration between regions have the same 4 intersection matrix, the approach proposes to consider further properties of the 4 intersection matrix in order to better distinguish among configurations. Such properties, that concern the boundary-boundary component of the 4 intersection matrix, are the boundary-boundary components, their types, dimensions, complement relations and sequences. Using such properties it is possible to identify any two equivalent topological relations between two regions and also it is possible to describe different levels of similarity, for example, by fixing the number and the types of the boundary-boundary components, but discarding their dimensions.

Another approach for the assessment of topological similarity, that seems a to suitable method for comparing multi-resolution datasets is presented in [EAT92]. The assessment of the similarity of the topological relations between two regions is based on the comparison of the content of their two 9 intersection matrices. In particular, the basic idea is to map the values empty or non-empty of the cells of the considered 9 intersection matrices onto the integers 0 and 1 respectively, and apply the integer subtraction.

$$\begin{gathered}\varnothing \text{ - } \varnothing = 0\text{-}0 = 0 \\ \neg\varnothing \text{ - } \varnothing = 1\text{-}0 = 1 \\ \varnothing \text{ - } \neg\varnothing = 0\text{-}1 = \text{-}1 \\ \neg\varnothing \text{ - } \neg\varnothing = 1\text{-}1 = 0\end{gathered}$$

By summing the result of the subtractions between corresponding cells in the matrices, we obtain a measure of the total difference between the topological relations; this measure is called *topology distance* and it is defined as follows.

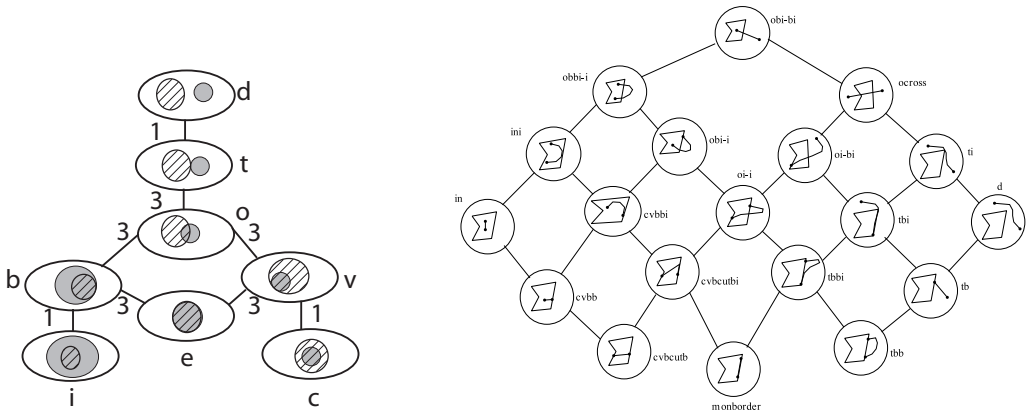
Definition 1 [Topology distance] Given two topological relations r_A and r_B the topology distance between them is the sum of the absolute values of the differences between corresponding entries in their 9-intersection matrices, M_A and M_B.

$$T_{r_A,r_B} = \sum_{i=\partial}^{-}\sum_{j=\partial}^{-} |M_A[i,j] - M_B[i,j]|.$$

□

The topology distance allows to define a partial order over a considered set of topological relations, thus, it is possible for a given topological relation to establish the closest relations among all relations of a considered set. Thus, considering the set of topological relations between regions, the closest topological relations can be represented as a graph, called *Closest-Topological-Relationship-Graph* (see Figure 2.8).

In [EM95], topology distance is used in two different models in order to assess similarity between two topological relations: the snapshot model and the smooth-transition model. The first model compares two topological relations without taking in account the transformations that have caused the change from one relation into the other relation. Thus, it is possible to compare two different existing maps without considering how such maps are created. This model uses the topology distance to evaluate the similarity of the topological relations between regions and lines; the conceptual neighbor graph derived is also presented (see Figure 2.8). On the other side, the second model is based on the concept of smooth-transition, that is two topological relations are conceptual neighbor if there exists a smooth transition from one to the other. Smooth transition means an infinitesimally small deformation that changes the topological relationships.



(a) (b)

Figure 2.8: Conceptual graph for topological relationships: (a) between two regions; (b) between region and lines.

2.4.2 Direction Consistency and Similarity

Similarly to topological relations, also direction relations can be modeled using a particular matrix (see Section 2.2). As a consequence, also for direction relations the proposed approaches for consistency and similarity checking are based on the comparison of the

contents of the matrices representing the considered relations. The characteristics of such approaches depend on the considered model for representing direction relations since it influences the structure of the matrix.

An approach for the similarity checking of the direction relations between two regions based on the D_9 model (see Section 2.2.2) is presented in [GE01, Goy00]. Such approach is an extension of the direction relation matrix called *detailed direction relation matrix* provides more details about directions among objects by considering additional criteria in non-empty cells of the matrix. The additional details concern the areal distribution throughout the direction tiles and the target object, and if a tile contains more than one disconnected part of the target object, the areal distribution within such tiles (Figure 2.9 (c)). The areal distribution captures how much of the target object falls into each tile. The basic idea under the approach is to assess similarity between two direction relations computing how to transform the matrix representing the first relation into the matrix representing the second relation.

Example 3 *Consider the direction relation represented in Figure 2.9 (b). The detailed matrix for such configuration has six elements zero and three non-zero which sum up to 1.0.*

$$D(A,B) = \begin{pmatrix} 0 & 0.05 & 0.45 & & \\ 0 & 0 & 0.50 & 0 & 0 \end{pmatrix}$$

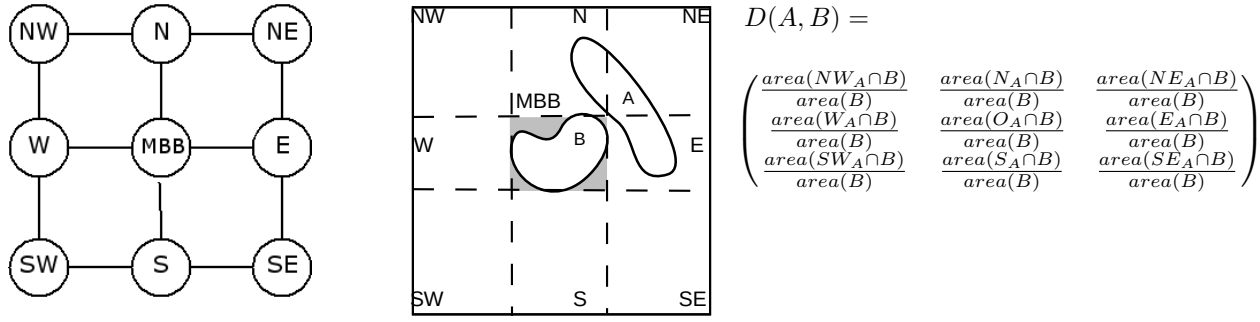
□

A detailed direction relation matrix with only one zero element is called a single-element direction relation matrix, and the corresponding relation is a single element (tile) direction relation. A direction relation matrix with more than one non-zero element is called a multi-element direction relation matrix and the corresponding relation represents a multi-element direction relation. Not all configurations of the detailed direction relation matrix represent a direction relation between two connected regions, it is necessary to establish some consistency constraints a matrix must satisfy to represent an existing direction relation. The constraints is based on a concept used in the categorization of neighboring cells in square tessellations, that is, two tiles are 4-neighbors if they are horizontally or vertically adjacent. Thus, any two non-empty intersections in a direction matrix have to be 4-connected, that is all pairs of non-empty cells are transitively 4-neighbors; otherwise, the representation in a direction-relation matrix would be inconsistent. The detailed direction matrix D^1 is compatible with the detailed direction matrix D^2 if the number of non-zero elements in D^2 is either equal to or more than the number of non-zero elements in D^1.

The first step for assessing similarity is the introduction of a distance measure between two direction relations, such that a zero value implies that both directions are identical. Given two detailed direction relations D_0, D_1, distance(D_0, D_1) > distance(D_0, D_2) means D_0 is more similar to D_2 than to D_1. The computation of the distance between two

single-element directions relies on the conceptual neighborhood graph for the nine single directions using the 4-neighborhood of the nine tiles. This graph has a vertex for each direction and an edge for each pair of directions that are horizontally or vertically adjacent (Figure 2.9(a)). The distance between two directions is the length of the shortest path between two directions in the conceptual neighborhood graph. The distance between two identical directions is zero, which is the shortest of all distances. The distance between the directions northwest and southeast is four, which is the maximum. The only other pair with the maximum distance is northeast and southwest. The distance between single element direction-relation matrices is used as the basis for the distance between multi-element direction-relation matrices. The distance between two detailed direction-relation matrices, D^0 and D^1, is the minimum cost for transforming matrix D^0 into D^1 by moving the non-zero elements of D^0 from their locations to the locations of the non-zero elements of D^1 along the conceptual neighborhood graph.

Such model is extended in [Goy00] to arbitrary pairs of point, line, and region objects.



(a) (b) (c)

Figure 2.9: Models for direction relations: (a) the conceptual neighbor graph for direction relations; (b) map representing a direction relation between two regions; (c) detailed direction relation matrix.

2.5 Geo-Spatial Query Processing

In the following, we introduce the main concepts related to geo-spatial query processing. In particular, in Section 2.5.1 we briefly introduce some of the well known geo-spatial query operators. In Section 2.5.2, we present some important index structures supporting and improving data access, then we discuss geo-spatial query processing for selection and join operators in Sections 2.5.3 and 2.5.4, respectively.

2.5.1 Geo-Spatial Queries

The first attempts to identify the characteristics of the fundamental geo-spatial queries have been presented in [AS91, G̈94, SA95]. It is nowdays recognized that the most important geo-spatial operators are substantially selection and join. The geo-spatial selection returns objects that satisfy a particular predicate with respect to a fixed geometric value (usually called *query object*). The geo-spatial join is defined as a binary operator that returns pairs of objects, each belonging to one input dataset, that satisfy a given condition.

Conditions of geo-spatial selection and join operators can refer either non-spatial or spatial attributes and may use geo-spatial relation as predicates. Depending on the used geo-spatial relation, selection and join operators can be classified into:

- *Topological operators.* Such operators return the (pairs of) geo-spatial objects that satisfy a topological relations (e.g., `overlap, touch, in`, etc...) with respect to a given object O, in case of selection, or with respect each other, in case of join. When considering selection and O is a rectangle, the query is usually called *range query*; when O is a point, it is called *point query* (see Figure 2.10(a)).

- *Distance-based operators.* Such operators return the (pairs of) geo-spatial objects that satisfy a distance relations (e.g., the distance between A and B is lower than a value d) with respect to a given object O, in case of selection, or with respect to each other, in case of join. The distance-based selection queries which retrieve all objects whose distance from O is equal or lower than a certain distance value (or between a maximum and a minimum distance value) are called *buffer query* (see Figure 2.10(b).

- *Direction-based operators.* Such operators return the (pairs of) geo-spatial objects that satisfy a given direction relation with respect to a given object O, in case of selection, or with respect each other, in case of join.

Another important geo-spatial selection operator is the nearest neighbor selection operator that, given a geo-spatial object O, usually a point, find all objects closest to O (see Figure 2.10(c)). A similar operator can be defined for join, retrieving for each object in the first dataset its nearest neighbor objects in the second dataset.

An important issue related to geo-spatial operators is their efficient computation. Indeed, given a map with n geo-spatial objects, the complexity of the naïve approach for both selection and nearest neighbor operator is in $O(n)$; on the other side, for join operator it is in $O(n^2)$. As a consequence, some methods for a more efficient computation are needed, in order to access and manage only the interesting geo-spatial objects and to early discard the non-interesting ones.

Since in our thesis we are interested especially on topological predicates, in the following, after a brief review of geo-spatial indexes, we discuss the more relevant approaches for processing topological selection and join operators.

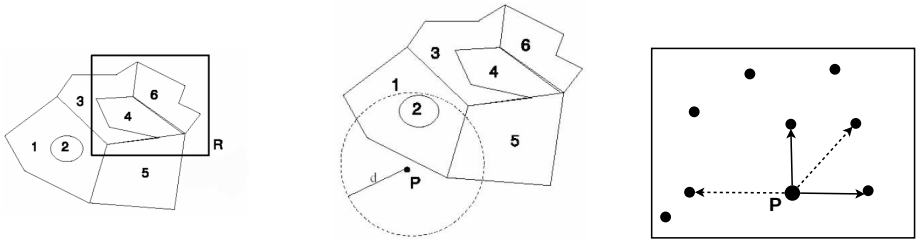


Figure 2.10: Different selection query types: (a) range query; (b) distance query; (c) nearest neighbor query.

2.5.2 Geo-Spatial Indexing

The need for an efficient access to geo-spatial data has resulted in the definition of time and space efficient access methods and geo-spatial indexes. The literature on geo-spatial data indexes is very extended, thus, a great variety of multi-dimensional access methods have been defined. In the following, we briefly describe only some of them. We refer the reader to [GG98, Sam05, Sam95] for further details.

The main characteristics of a geo-spatial index is to organize the reference space and the objects in some way so that only parts of the space and a subset of the objects need to be considered in order to answer a query. The geo-spatial index is generally a hierarchical data structure based on a recursive decomposition of the space and the assignment of objects to the obtained space portions.

More in details, a hierarchical data structure is a tree structure where each node corresponds to a region of the reference space and data objects are stored into leaves. An internal node region covers the regions of its subnodes and each node may or may not overlap other nodes, depending on the index type. Each node is typically stored in one page of external memory, whose size is determined by the underlying DBMS.

Due to the arbitrary complexity of real geo-spatial objects, it is not advisable to build up an index using the complete geometric description of the objects; thus, objects are usually approximated. One of the most used approximation is the *minimum bounding rectangle (MBR)*, a X-Y parallel rectangle that fully encloses the geometry of a geo-spatial object. Indexes based on MBRs only manage rectangles and maintan only a reference to the exact object geometry. As a consequence, any processing involving the index may produce a

superset of the actual result (*candidate solutions*), corresponding to the MBRs of objects that "might" satisfy the query (*filter step*). A *refinement step* is then needed in order to discard false hits. This step requires access to the exact object geometry.

Geo-spatial indexes can be divided into two main groups, depending on whether the space decomposition depends or not on objects to be indexed. In the first case, indexes are called *data driven*, in the second case *space driven*.

Data driven indexes can be divided into three different groups: (i) transformation-based, (ii) overlapping-based, and (iii) clipping-based approaches. Transformation-based indexes transform each geo-spatial object into a high-dimensional point and then rely on some data-driven data structure for point indexing. An example of this kind of indexes is the LSD tree [Sam05]. Overlapping-based approaches divide the space into regions which may overlap. Each object is then assigned to one of such regions. Examples of such type of indexes are the R-tree [Gut84], and the R*-tree [BKSS90] (see Figure 2.11 (a) and (b) respectively), An R-tree is a hierarchical data structure, corresponding to a hierarchy of nested d-dimensional rectangles, organized in a height-balanced data structure where each node corresponds to a disk page. Each node also corresponds to a rectangular region, defined as the MBR that fully contains the regions of its children. The leaf nodes contain the reference to the geometry of the indexed objects. Finally, clipping-based approaches divide the space into non-overlapping regions. The R^+-tree [SRF87] is an example of a clipping-based approach. It mainly differs from the R-tree in the insertion algorithm, which is able to optimize the area occupied by the internal nodes (see Figure 2.11 (c)). Since the importance of R-tree grows up in recent time, several different implementations have been developed for it. Most of them can be found at the R-tree portal [rtr].

In space-driven indexes, the reference space is divided into cells, independently from objects distribution. Objects are associated with cells using some particular criteria. This group of indexes includes quadtrees, octrees, k-d-trees, their variants, and several other data structures. Since we do not deal with such types of indexes in this PhD thesis, we refer the reader to [Sam05] for additional details.

2.5.3 Query Processing for Selection Operator

Several efficient methods have been developed exploiting geo-spatial indexes for guaranteeing a fast answer to geo-spatial query selection. Most approaches focus on topological conditions, especially intersection and containment. Indeed, among geo-spatial queries, those based on geo-spatial and more precisely topological relations are used in an increasing number of applications, including spatial and geographical applications, spatial reasoning, image and multimedia databases, and cognitive sciences. Surprisingly, although the importance of topological relations is well known, most of the existing works on geo-spatial

processing has concentrated on fixed-domain topological relations, especially on those for pairs of regions and only few proposals exist for efficiently execute queries based on generic topological relations. In the following, we briefly review the query processing approaches for topological selection relying on the usage of R-trees or their variants, since they are most relevant for the content of this PhD thesis.

When considering intersection or containment, query processing based on R-trees is quite simple since, in order to determine objects that intersect or are contained in a given query object O, it is sufficient to discard all (internal or leaf) MBRs which do not intersect or contain the MBR of O. In both cases, the query is processed in two steps: (i) *filter step*, which finds all objects whose MBR intersects/contains the query rectangle; (ii) *refinement step*, that, for those objects, check whether they really satisfy the query condition.

When a generic topological predicate is considered, the processing is not so easy. Indeed, it is no more true that if a pair of MBRs satisfy a given topological relation, such relation is also satisfied by their children. In this case, processing relies on the usage of *compatibility relations*, described in [PTSE95] for regions. Indeed, there exists a consistent mappings between the topological relation satisfied by pairs of MBRs, associated with leaf entries, and the topological relation between the objects they represent. For example, from Table 2.1, we can see that, when two leaf MBRs satisfy relation `touch`, their corresponding objects may satisfy relations `disjoint` or `touch`. The reverse mapping from the relation satisfied by exact geometries onto relations satisfied by their MBRs is straightforward (see Table 2.2). In Tables 2.1 and 2.2, `o, t, c, i, v, b, d, e` denotes relations `overlap, touch, contain, in, cover, coveredby, disjoint,` and `equal`, respectively. Based on the relation satisfied by a (both leaf or internal) node MBR, say p', and the MBR of the query object, say q', it is also possible to determine the set of relations that may be satisfied by MBRs associated with the node ancestors, say P, and q' (see Table 2.3). The proposed tables can be used for processing selection queries based on generic topological relations as follows. The usual R-tree processing is started. However, visit of a node is pruned if, based on Tables 2.1, 2.2, and 2.3, the relation between its MBR and the MBR of the query object q' does not allow exact geometries referenced in the leaves to satisfy the query relation. As an example, Figure 2.12, assuming that an MBR p' touches q', shows the topological relationships that may be satisfied by an ancestor MBR P with respect to q'.

We recall that a similar approach is applied in [PTS94] to define query processing techniques for direction relations, relying on the usage of R-trees or their variants.

MBRs relations	Exact geometries relations
d	d
t	d ∨ t
o	d ∨ t ∨ o
i	d ∨ t ∨ o ∨ i ∨ b
b	d ∨ t ∨ o ∨ i ∨ b
c	d ∨ t ∨ o ∨ c ∨ v
v	d ∨ t ∨ o ∨ c ∨ v
e	e ∨ o ∨ b ∨ v

Table 2.1: Mapping of MBR relations onto exact geometries relations.

Exact geometries relations	MBRs relations
d	d ∨ t ∨ o ∨ i ∨ b ∨ c∨ v
t	t ∨ o ∨ e ∨ i ∨ b ∨ c∨ v
o	o ∨ i ∨ b ∨ c∨ v
i	i ∨ b
b	i ∨ b∨ e
c	c ∨ v
v	c ∨ v∨ e
e	e

Table 2.2: Mapping of exact geometries relations onto MBR relations.

2.5.4 Query Processing for Join Operator

Geo-spatial join is one of the fundamental geo-spatial operators since it is involved in several common queries. Unfortunately, due to the complexity of geo-spatial data, its computation suffers of high execution costs. Thus, a lot of work of the geo-spatial research community focused on the development of techniques to improve performance of geo-spatial join. As a consequence several algorithms considering both indexed and non-indexed geo-spatial datasets have been developed. In the following, we briefly review the basic techniques for processing the join operator on geo-spatial datasets indexed by R-trees or their variants, since this is the case which is more relevant for the topics addressed by this PhD thesis.

Similarly to selection, query processing techniques for join have been proposed mainly for intersection and containment topological relations. In [BKS93], three basic techniques, based on the usage of R-trees, for computing geo-spatial join are presented and compared:

- *MBR-spatial-join*, that computes all pairs of object identifiers ($Id(a)$, $Id(b)$) such that their MBRs intersect;
- *ID-spatial-join*, that computes all pairs of object identifiers ($Id(a)$,$Id(b)$) such that their object geometry intersect;
- *object-spatial-join*, that computes the geo-spatial intersection between any pairs of objects.

Relation between p' and q'	**Relations between the MBRs of ancestors of the node associated with p' and q'**
e	e ∨ v ∨ c
c	c
i	o ∨ v ∨ c∨ i∨ e∨ b
v	c ∨ v
b	o ∨ v ∨ c∨ e∨ b
d	d ∨ t ∨ o∨ v∨ c
t	t ∨ o∨ v∨ c
o	o ∨ c ∨ v

Table 2.3: Mapping of relations between $p' = MBR(n)$ and $q' = MBR(O)$ onto relations between $P = MBR(n')$ and $q' = MBR(O)$, where n' is an ancestor of n.

Since the MBR-spatial-join can be used to implement the filter step of the other two techniques, in the following we describe it in details. The approach is based on the following property: since the MBR of an internal node n represents the minimum bounding box for all the MBRs contained in the subtree rooted by n. Thus, given two R-trees nodes N_a containing entries $E_1^a, ..., E_n^a$ and N_b containing entries $E_1^b, ..., E_m^b$ respectively of the R-trees R_a and R_b. If the MBRs of N_a and N_b do not intersect, there will be no pairs of their entries that intersect. Thus, the pair (N_a, N_b) can be discarded. Otherwise, there might be a pair of intersecting entries belonging to N_a and N_b and therefore they should be further visited.

The basic algorithm for MBR-spatial-join considers a pair of R*-trees with the same height (see Algorithm 1). For R-trees with different height, the basic algorithm does not change for pairs of both internal/leaf nodes. For pairs composed of one intermediate node, say N_a, and one leaf node, say L_b, the idea is to perform a geo-spatial selection query between the subtree rooted by N_a using the MBRs of L_b as query rectangle.

The main drawback of the MBR-spatial-join is due to the high number of pairs visited. Various optimizations have therefore been proposed in order to reduce such number. The first optimization is based on the following property. Given two intermediate entries N_a and N_b, such that $MBR(N_a) \cap MBR(N_b) \neq \emptyset$, let $Rect_{RS} = MBR(N_a) \cap MBR(N_b)$. Then, only the entries E_a of N_a and E_b of N_b, that intersect $Rect_{RS}$ may have common intersection. All the other pairs can be discarded. The second optimization orders the entries of the nodes of the considered R-trees according to their geo-spatial location, and then using a plane sweep approach to find the pairs of intersecting entries.

A complete survey and analysis of geo-spatial join techniques is presented in [JS07]. This is not just a review of the state-of-the-art in the design of geo-spatial join algorithm, but it aims to compare the different existing approaches pointing out the strength and weakness of each of them. To simplify the discussion, only two-dimensional data are considered and the target is to compute the pairs of intersecting objects. Specialized algorithms are also considered to perform geo-spatial joins that run in parallel and distributed environments.

Algorithm 1 Geo-Spatial Join Algorithm for the R-trees R_a, R_b with same height.

```
1: INPUT
2: node N_a of R-tree R_a, node N_b of R-tree R_b
3: METHOD
4: for all entries E_a ∈ N_a do
5:     for all entries E_b ∈ N_b with MBR(E_b) ∩ MBR(E_a) ≠∅ do
6:         if R is a leaf page then
7:             output(E_a,E_b)
8:         else
9:             SpatialJoin(E_a,E_b)
10:        end if
11:    end for
12: end for
```

Three main algorithms are considered:

1. The first algorithm assumes that both datasets are indexed using a hierarchical data structure, such as the R-tree or its variants.

2. The second algorithm considers indexes simply as partitioned datasets, where the data pages of indexes are the partitions.

3. The third algorithm transforms geo-spatial objects into multi-dimensional points and use data structures for them in order to perform the join.

The first approach is presented in Algorithm 2, which implements a synchronized traversal of the corresponding levels in the two R-trees with the same height, at the same time.

Algorithm 2 Generic synchronized traversal of two R-trees R_a, R_b with the same height.

```
1: INPUT
2: rootA the root of R_a, rootB the root of R_b
3: METHOD
4: priorityQueue := CREATE_PRIORITY_QUEUE()
5: priorityQueue.ADD_PAIR(rootA, rootB)
6: while NOT priorityQueue.EMPTY() do
7:     nodePair := priorityQueue.POP()
8:     rectanglePairs := FIND_INTERSECTING_PAIRS(nodePair)
9:     for all p ∈ rectanglePairs do
10:        if p is a pair of leaves then
11:            REPORT_INTERSECTIONS(p)
12:        else
13:            priorityQueue.ADD_PAIR(p)
14:        end if
15:    end for
16: end while
```

The algorithm starts from the two root nodes of the indexes, `rootA` and `rootB`, and finds intersections between the children of `rootA` and `rootB` using the `FIND_INTERSECTING_PAIRS` function. Intersecting children pairs are added to the priority queue `priorityQueue`, and

checked for intersecting children in later iterations. If the two nodes are leaves, then the `REPORT_INTERSECTIONS` function is used to compare the leaves and report any pair of intersecting objects.

Different algorithms can be obtained by different implementations of function `ADD_PAIR`, which adds a pair to the priority queue. In [JS07], some variations are proposed with the aim of minimizing disk accesses. The first variation performs a breadth-first traversal of the tree, where priority is given to higher-level node pairs. In this way, all of the nodes at one level of the indexes are examined before any node in the next level. Another variation further sorts the pairs at each level by considering a secondary sort, based on various heuristics (for example, clustering pairs on one datasets nodes).

One of the drawbacks of the breadth-first approach is that, as the algorithm proceedes, the priority queue can grow extremely large and portions of it might need to be kept in external memory. Another approach uses a depth-first visit in which all subnode pairs for a given node pair are processed before proceeding to the next node pair at the same level. In this case, priority is given to lower level node pairs. As with the breadth-first approach, heuristics can be used to reduce I/O costs by using secondary ordering for the considered pairs.

The third main algorithm considered in [JS07] assumes that each geo-spatial object is transformed into a multi-dimensional point. Indeed, one problem with objects with extent, as rectangles, is that objects may not neatly fit into a partition, thus the objects must be replicated into multiple partitions or the partitions must overlap, as in an R-tree. Transformation methods avoid this problem by transforming objects into multi-dimensional points in a higher-dimensional space. For example, a two-dimensional rectangle can be transformed into a point in the four-dimensional space by using the coordinate values of the center point, half of the width, and half of the height as the four values representing the rectangle. Then, indexes can then be used to create a partitioning of the multi-dimensional point dataset.

2.6 Relaxed Geo-Spatial Query Processing

The concept of *query relaxation* has been introduced in Information Retrieval in order to allow the retrieval of a result also in very heterogeneous contexts or when the characteristics of data we are looking for are not completely known and this may lead to imprecise query specification. The main idea of query relaxation is to modify the query in order to stretch or shrink the result set. Query relaxation approaches can be classified depending on their scope into *query rewriting* and *preference-based* approaches.

Under query rewriting approaches, the query is rewritten using less or more strict operators,

in order to get a larger or smaller answer set. An example of query rewriting technique is given by the geo-spatial relaxed topological selection operators and nearest neighbor operators proposed in [BBC+06]. Relaxation in this case is applied by considering during the execution topological relations whose similarity with respect to that specified in the query is higher than a given threshold, specified by the user in the query. In that, they are similar to the selection operator based on the Threshold semantics we define in this paper. However, differently from [BBC+06], we do not introduce it through rewriting but we provide specific query processing algorithms. Nearest neighbor operators, if the query condition is undefined for the spatial data at hand (for example, the user asks for all the rivers that cross a street, when rivers and streets are polygons and therefore relation cross is not defined) substitute the query relation with the most similar ones defined for the considered objects. Such operators can however return an empty answer when no object in the dataset satisfies the set of relaxed topological relationships. Thus, they differ from Best Fit semantics we propose in this paper. Additionally, no processing strategy was proposed for their execution.

In *Preference-based approaches*, user or system preferences are taken into account in order to generate the result, with the aim of providing best results first. Two main operators have been proposed based on user preferences: top-k and skyline operators.

The aim of the top-k operator is to restrict the number of returned results to a fixed number (k), based on some ranking function. In the geo-spatial context, top-k operators return the first k objects that satisfy a given condition, based on a preference function computed over spatial and non spatial attributes.

The aim of the skyline operator is to return only the best matches to the specified query, without using a specific ranking function. Best matches can be computed by providing preferences in terms of sets of attributes considered relevant for the ranking. Given a set of points, each representing a list of values for the list of relevant attributes, the skyline contains the points that dominate all other points. A point A dominates a point B if it is better in at least one dimension and equal or better in all the others, by considering a specific scoring function [BKS01b].

In the following we briefly discuss some existing approaches for relaxed query processing of selection (Section 2.6.1), join (Section 2.6.2) operators in the geo-spatial context.

2.6.1 Relaxed Query Processing for Selection

While several approaches have been proposed for relaxed query processing of selection operations ([CG99, BCG02, KLTV06]), while only few of them refer to the geo-spatial context. Among them, proposals for re-defining the concept of skyline and top-k queries in the geo-spatial contexts have been proposed.

More precisely, the concepts of relaxation, domination and skyline are adopted in the geo-spatial context to define the concept of spatial skyline queries (SSQ) [SS06]. Given a set of data points P and a set of query points Q in a d-dimensional space, a spatial skyline query retrieves those points of P which are not dominated by any other point in P. Dominance is defined with respect to the values of the points for a set of spatial derived attributes. Given the set of points Q, Euclidean distances to query points in Q is a spatial derived attribute for the points in Q that may be considered for defining dominance.

The naive brute-force search algorithm for finding the spatial skyline of P given a query set Q requires to examine all points in P against each other. For each point p, $|Q|$ distances $D(p, q_i)$ are computed and compared against the corresponding distances of other points.

Some approaches for an efficient execution of SSQ algorithms have therefore been proposed, that, by using some specific geometric structures (e.g., Voronoi diagram, Delaunay graph, and convex hull), allow us to avoid a significant number of distance computations with respect to the naive approach.

A geo-spatial selection top-k operator has been proposed in [YDMV07] for points. It considers both: (i) spatial ranking, by ordering objects according to their distance from a reference point; (ii) non-spatial ranking, by ordering objects according to an aggregate function on their non-spatial values. Based on such ranking, a top-k preference query retrieves the k points in a set of interesting points with the highest score for the considered ranking.

2.6.2 Relaxed Query Processing for Join

Due to the importance and complexity of the join operator, several approaches have been proposed for join relaxation in non-geo-spatial contexts [KLTV06, IAE03]. Similarly to selection, also in this case only few approaches consider relaxing geo-spatial join. Among them [ZPZL05] propose a top-k geo-spatial join operator. The proposed operator retrieves the k objects in dataset A or B that intersect the maximum number of objects from the other dataset. In order to efficiently compute such join, an R-tree, or one of its variants, is used and visited using a branch and bound approach.

The branch-and-bound approach uses the following concepts: (i) the *pruning condition*, which excludes subtrees that cannot lead to better results, than those already found; (ii) the *key*, that determines the order by which the qualifying subtrees are visited so that good solutions are identified as early as possible.

Values for the pruning condition and the key are assigned to any entry e of an internal node of R-tree R_a, based on two specific metrics. One metric coincides with the upper bound of the number of objects in a subtree rooted by e. The other metric, denoted

Algorithm 3 Top-k GEo-Spatial Join Algorithm for two R-trees R_a, R_b with same height.

```
INPUT
root of R_a, root of R_b
METHOD
join R-tree R_a, R-tree R_b to get intersecting pairs (e_a, e_b)
for all entry e that appears in a pair do
    build e.IL, compute e.count and insert < e, e.count, e.H > to heap H (sorted by e.count)
    while number of reported objects <k do
        e=deheap(H)
        if e is a leaf entry then
            report (< e.id, e.count, e.H >)
        else
            for all e_i ∈ e.IL do
                join n and n_i
                for all each intersecting pair e', e'_i do
                    add e' to e'.IL
                end for
                compute e'.count
            end for
            if e'.count > pruning condition then
                insert < e', e'.count, e'.H > to H
                if e' is a leaf entry then
                    update pruning condition
                end if
            end if
        end if
    end while
end for
```

by $count(e)$, corresponds to (the upper bound of) the number of objects indexed by R_b that intersect the rectangle corresponding to e. The metric *count* provides a termination condition, since it is not necessary to visit entries whose *count* is smaller than that of the top-k results already found. It also provides a visit order, since we first visit entries with high *count* based on the intuition that they are likely to lead to interesting results. The basic algorithm presented in [ZPZL05], considering two R-trees with the same hight, is shown in Algorithm 3.

The result computed by the top-k geo-spatial join has some similarity with the problem of finding the K-closest pairs of objects between two geo-spatial datasets. Indeed, also in this case it is necessary to relate all the objects of one dataset with all the objects of the other dataset, returning those pairs satisfying a certain properties. This approach combines geo-spatial join and nearest neighbor query, and it is denoted by *K-Closest Pair Query (K-CPQ)*. In [CMTV00], a processing techniques for K-CPQ queries over point datasets is provided. In order to prune R-tree subtrees, two metrics are defined, based on the following considerations (see Figure 2.13): (i) the MBR of each internal node contains all the points stored in its descendant nodes; (ii) since the MBR of a given set of points is the minimum rectangle that contains all such points, at least one point of the considered set is located on each edge of each MBR associated with each internal node.

Given two MBRs M_P and M_Q of two R-trees R_P and R_Q, and denoting by r_i (s_j) the

vertexes of M_P and M_Q, respectively, the defined metrics relies on the following values:

- $MINDIST(r_i, s_j)$ as the minimum distance between two points falling on r_i and s_j;
- $MAXDIST(r_i, s_j)$ as the maximum distance between two points falling on r_i and s_j.

The metrics are the following (see Figure 2.13):

- $MINMAXDIST(M_P, M_Q) = min_{i,j} MAXDIST(r_i, s_j)$ is the upper bound of the distances of the pair of points belonging to the considered edges.
- $MAXMAXDIST(M_P, M_Q) = max_{i,j} MAXDIST(r_i, s_j)$ is is the maximum distance of any two points belonging to the considered MBRs
- $MINMINDIST(M_P, M_Q) \leq d(p_i, q_j) \leq MAXMAXDIST(M_P, M_Q)$ and $d(p_i, q_j) \leq MINMAXDIST(M_P, M_Q)$, where $d(p_i, q_j)$ denotes the distance between two points p_i, q_j of the considered MBRs M_P, M_Q.

The considered metrics allows the definition of various pruning conditions, to be used to prune subtrees of the R-trees which are not involved in answer generation. In particular, various recursive and non-recursive algorithms are presented for finding the 1-closest pair, considering R-trees of the same height. The basic algorithm consists of the following steps: (i) start visiting the two R-trees from their roots and set the minimum distance T to ∞; (ii) for each pair of internal nodes, we check if it satisfies the pruning condition defined using the metrics above, if the condition is satisfied the visit is propagated downwards recursively for every possible pairs of nodes entries; (iii) as soon as a pair of leaves is reached, calculate the distance of each possible pair of points. If this distance is smaller than T, update T.

The case of different heights of the R-trees are resolved using these two methods.

- In the approach "$fix-at-root$" the downward propagation stops in the R-tree with lower level node, while propagation in the other tree continues, until a pair of nodes at the same level is reached. Then the visit continues in a synchronous way.
- In the approach "$fix-at-leaves$" the downward propagation stops in the R-tree with lower level node when a leaf is reached, while propagation in the other tree continues.

Finally, an heap is used to hold the K-closest pairs ordered in descending distance, thus the pairs with the largest distance are on the top of the heap. Such approach is presented in [CMTV00, CMTV04].

Symbol	Description
ft, f	feature type, feature
OBJ	the set of geo-spatial objects defined by OGC
DIM	{0,1,2} the set of possible dimensions for the objects in OBJ
$d(o)$	the dimension (in DIM) of object o
$d(ft, MS)$	the dimension (in DIM) of feature type ft in the map schema MS
$d(f, M)$	the dimension (in DIM) of feature f in map M
$ext(ft, M)$	the set of features associated with a feature type ft inside a map M

Table 2.4: Symbols used in the rest of the thesis.

2.7 Thesis Background

The contributions of this PhD thesis rely on several models and approaches discussed in this chapter. In the following, we briefly summarize the background models and methods used to develop the proposed techniques.

- *Geo-spatial data model.* We consider geo-spatial data represented according to the vector model and feature type-based maps (see Section 2.3). We assume that values for the spatial attribute are modeled according to the OGC Simple Features Specification for SQL [OGCb]. Maps can be multi-resolution, as explained in Section 2.3.
- *Geo-spatial relationships.* We consider both topological and direction relationships. For them, specific models are provided, relying on the 9-intersection model [CDFVO93] and on the approach proposed in [SK04, SK02], respectively.
- *Type of geo-spatial index.* We assume that instances of a feature type in a given map are indexed using an R-tree or one of its variants.
- *Considered operators.* We consider both relaxed selection and join operators. Relaxation is applied in a trasparent way by the system.
- *Query processing.* The query processing techniques we propose assume that geo-spatial data is indexed as explained before. They rely on the branch-and-bound approach presented in Section 2.6.2.

In Table 5.2, we present the notation used in the rest of this thesis.

2.8 Summary

In this chapter, we reviewed the existing work about the concepts and notions we exploit in this thesis, highlighting their advantages and disadvantages.

First of all we presented the models for a the representation of geo-spatial data, their associated geo-spatial relationships (topological, direction, and metric) and the approaches for the assessment of consistency and similarity of geo-spatial maps using geo-spatial relationships.

Then, we discussed the critical issues related to geo-spatial query processing; in particular, we presented the existing methods for an efficient processing of geo-spatial selection and join operators both in a precise and approximat context.

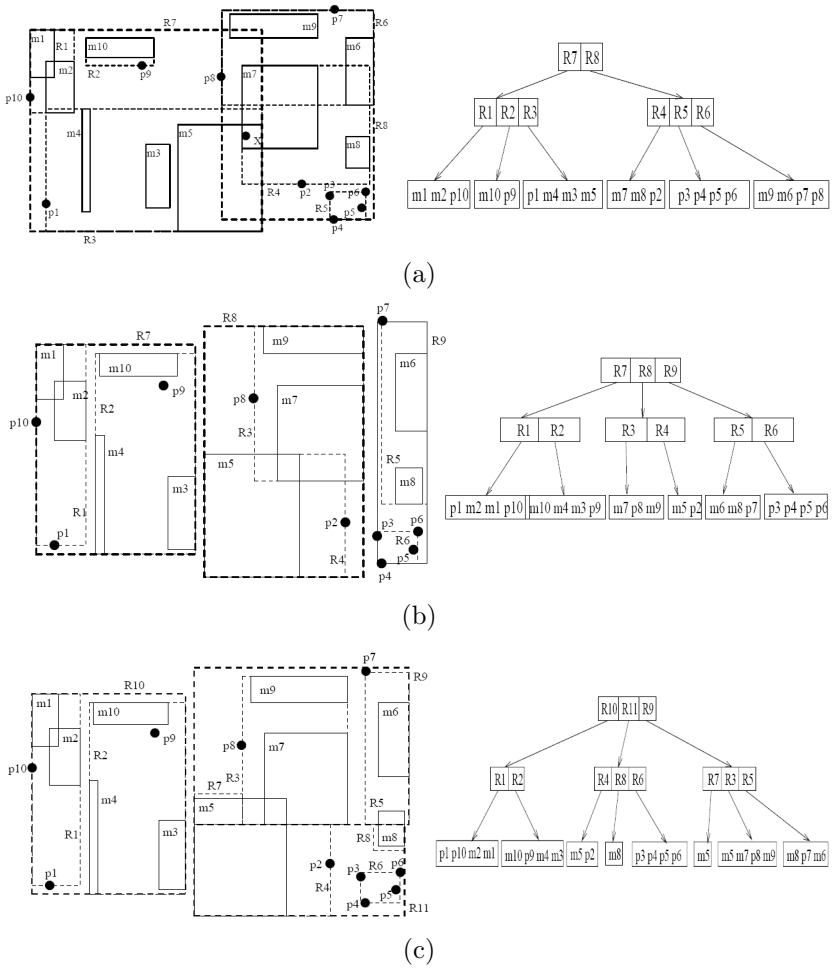


Figure 2.11: The same dataset represented using: (a) R-tree; (b) R*-tree; (c) R$^+$-tree [GG98]

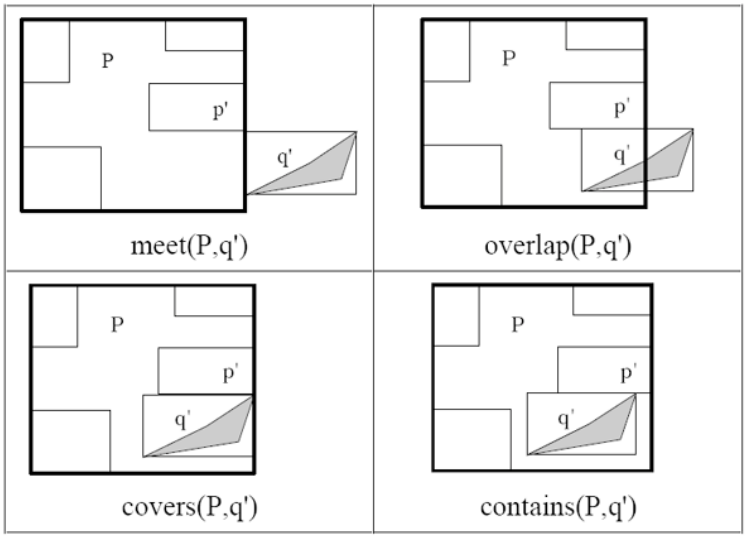


Figure 2.12: Possible topological relations between the MBR of an intermediate node P and q', if p' `touch` q'.

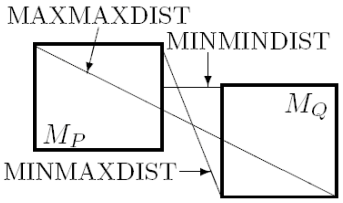


Figure 2.13: Example of the metrics applied to pairs of MBRs.

Chapter 3

Multi-resolution Geo-Spatial Datasets Comparison Using Qualitative Information

As already discussed in Chapter 2, nowadays, very often new geo-spatial applications, especially in distributed environments, need to cope with multi-resolution geo-spatial datasets. In this context, a relevant issue concerns the evaluation of consistency and similarity among such datasets. Even if various approaches for consistency checking have already been defined (see Chapter 2), they often assume that maps are not multi-resolution. Such approaches do not always exploit geo-spatial relations in performing the consistency analysis.

The aim of this chapter is to make one step towards overcome such limitations by presenting a framework for the evaluation of consistency and similarity between multi-resolution maps based on geo-spatial relations. Due to their importance in GIS applications, we mainly consider topological and direction relations. For them, we first propose a representation model, obtained by extending some previously defined models in order to cope with multi-resolution datasets. Then, we present some similarity functions, to be used for consistency and similarity checking of multi-resolution datasets, as well as for query relaxation. Based on the proposed functions, we finally provide an overall framework for consistency checking based on qualitative information.

The remainder of this chapter is organized as follows. In Section 3.1, the reference geo-spatial model for multi-resolution geo-spatial datasets is presented; a reference model and two different similarity functions for topological relationships are presented in Section 3.2 while a similar model and function for direction relationships are then presented in Section 3.3. Finally, in Section 3.4, the defined functions are used to define a formal framework for consistency checking.

3.1 The Reference Multi-resolution Geo-Spatial Data Model

As pointed out in Section 2.7, we consider geo-spatial data represented according to the vector model and organized into maps (see Section 2.3). We assume that values for the spatial attribute are modeled as simple geometries according to the OGC Simple Features Specification [OGCc]. Maps can be multi-resolution, according to the feature type-based map model, introduced in Section 2.3.

Usually, in the geo-spatial context, generalization means the trasformation of an existing map into a more general one through some operations of generalization, that perform a reduction of the number and of the details of the objects in the map. By considering our reference model, we define a *generalization (specialization)* relation between two given maps based on the dimensions assigned to feature types contained in them.

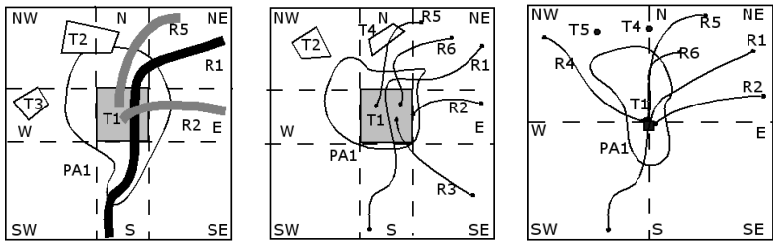
Definition 2 (*Generalization and specialization*) Let MS_1 and MS_2 be two map schemas and $ft \in MS_1 \cap MS_2$. MS_1 generalizes MS_2 (MS_2 specializes MS_1) with respect to ft, if $d(ft, MS_1) < d(ft, MS_2)$. MS_1 generalizes MS_2 (MS_2 specializes MS_1) if the following conditions hold:

- $\exists ft \in MS_1 \cap MS_2$, MS_1 generalizes MS_2 (MS_2 specializes MS_1) with respect to ft;
- $\forall ft \in MS_1 \cap MS_2$, $d(ft, MS_1) = d(ft, MS_2)$ or MS_1 generalizes MS_2 (MS_2 specializes MS_1) with respect to ft. □

Example 4 *Consider three distinct maps M_1, M_2, and M_3, sketched in Figure 3.1, representing roads (identified by R_i), towns (identified by T_i) and pollution areas (identified by PA_i) with different dimensions respectively (2, 2, 2) in M_1, (1, 2, 2) in M_2, and (1, 0, 2) in M_3. Then we can say that M_3 generalizes both M_1, M_2 and that M_2 generalizes M_1. On the other side, M_1 specializes both M_2, M_3 and M_2 specializes M_3.* □

3.2 Topological Relationships

In the following, we first present a formal model for topological relations; then, two similarity functions for comparing two topological relationships, possibly defined over multi-resolution pairs of features, under different environments are provided.



M_1 M_2 M_3

Figure 3.1: Specialization and generalization of maps.

3.2.1 The Model

The model we consider for the representation of topological relationships is a variation of the *extended 9-intersection model*, defined in [CDFVO93] and briefly presented in Section 2.2.1.

This model introduces the following set of topological relationships: $\mathcal{T}$ = {`disjoint(d), touch(t), in(i), contain(c), equal(e), cross(r), overlap(o)` }. While such relationships are at the basis of query processors of well known geo-spatial systems, they are not able to distinguish between different types of containment, which could be relevant for some applications, as described in the following example.

Example 5 *Suppose to consider a map with the rivers of Liguria, a region of Italy, and suppose we want to know only the rivers in Liguria that flow into the sea. A river satisfying this condition is such that it is contained in Liguria and touches its boundary. To model such situation, relation* `in` *can be used. However, such relation is satisfied by all Liguria rivers, indipendently on the fact they flow or do not flow into the sea. Indeed, using* $\mathcal{T}$ *there is no way to distinguish between these two types of rivers.* □

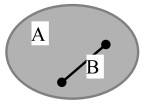
Based on the previous example, and similarly to what has been done in [PTSE95], we decided to refine the *extended 9-intersection model* by inserting two new relations, `cover` and `coveredby`, and redefining semantics for relations `in` and `contain`, in order to still obtain a mutually exclusive set of topological relations. More precisely, two objects satisfy relation `in` or `contain` if the interior and the boundary of one object are contained in the interior of the other; on the other hand, if the boundary of one object touches the boundary of the other, we say that the two objects satisfy relations `cover` or `coveredby`.

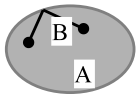
The set of topological relations considered from now on is therefore the following: $\mathcal{T}$={`touch (t), in(i), cross (r), equal (e), overlap (o), disjoint (d), contain (c), cover (v), coveredby (b)`}.

The semantics of the considered set of topological relations is provided in Table 3.1. The second column presents the formal definition of the relation considering the intersections among the whole features, their interiors and their boundaries. The third column presents the pair of dimensions for which the relation is defined, since some relations, such as `cross`, are defined only for some particular pairs of dimensions. Since each main relation groups a set of 9-intersection matrices, the last column presents the pattern representing all the 9-intersection matrices that are grouped in the topological relation pointed out in the first column. The pattern is a string $P_{1,1}P_{1,2}P_{1,3} - P_{2,1}P_{2,2}P_{2,3} - P_{3,1}P_{3,2}P_{3,3}$ where element $P_{i,j}$ corresponds to cell (i, j) in the 9-intersection matrix. If $P_{i,j} = *$ then this position is not relevant in defining the topological relation, if $P_{i,j} = F/T$ means that the intersection is (or is not empty), $P_{i,j} \in \{1, 2, 3\}$ means that the intersection has the specified dimension x.

Example 6 *Consider the topological relation* $\mathtt{c}_{2,1}$ *corresponds to the following two 9-intersection matrices:*

$\mathtt{contain}^1_{1,2}=$ $\mathtt{contain}^2_{1,2}=$





$$\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix} \quad \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

By considering the pattern of `contain` *in Table 3.1, it is easy to verify that the above 9-intersection matrices satisfy such pattern. The pattern for* `contain` *is T*T-*FT-FFT. Both the matrices have non-empty intersections in the cells at position (1,1), (3,3) corresponding to value T of the pattern at positions* $P_{1,1}$, $P_{3,3}$. *The matrices have empty intersection at position (2,2) corresponding to value F of position* $P_{2,2}$ *of the pattern. Then, the matrices have non-empty intersection at position (2,3) corresponding to value T of position* $P_{2,3}$ *of the pattern. Finally, the matrices have empty intersection at positions (3,1), (3,2) corresponding to value F of positions* $P_{3,1}$, $P_{3,2}$ *of the pattern; finally, the non-empty intersection of the matrices cells (3,3) correspond to position* $P_{3,3}$ *of the pattern.* □

In the rest of this thesis, we use the following notations:

- θ_{d_1,d_2} the topological relation $\theta \in \mathcal{T}$ defined for d_1, d_2 in DIM (e.g., $d_{2,2}$ means `disjoint` between two regions);

Name	Definition	Object dimension	Pattern of the 9-int. matrix
`disjoint (d)`	$f_1 \cap f_2 = \emptyset$	d_1/d_2, with $d_1 \neq 0$, $d_2 \neq 0$,	$FFT - FFT - T * T$
		d_1/d_2, with $d_1 = 0$ or $d_2 = 0$,	$FFT - FFF - T * T$
`touch (t)`	$(f_1^\circ \cap f_2^\circ = \emptyset) \wedge (f_1 \cap f_2) \neq \emptyset)$	2/2, 2/1, 1/2, 1/1	$F * * - * T * - * * T$
		2/1, 2/0, 1/1, 1/0	$F * * - TF * - * * T$
		1/2, 1/1, 0/1, 0/2	$FT * - FF * - * * T$
`in (i)`	$(f_1 \cap f_2 = f_1) \wedge (f_1^\circ \cap f_2^\circ) \neq \emptyset)$	2/2, 1/1, 1/2, 0/2, 0/1	$T * F - * FF - TTT$
contain (c)	$(f_1 \cap f_2 = f_2) \wedge (f_1^\circ \cap f_2^\circ) \neq \emptyset)$	2/2, 2/1, 2/0, 1/1, 1/0	$T * T - * FT - FFT$
`equal (e)`	$f_1 = f_2$	2/2, 1/1,	$TFF - FTF - FFT$
		0/0	$TFF - FFF - FFT$
`cross (r)`	$d(f_1^\circ \cap f_2^\circ) = (max(d(f_1^\circ), d(f_2^\circ)) - 1) \wedge (f_1 \cap f_2) \neq f_1 \wedge (f_1 \cap f_2) \neq f_2$	1/2, 2/1	$T * T - * * * - T * T$
		1/1	$0 * T - * * * - T * T$
`overlap (o)`	$d(f_1^\circ) = d(f_2^\circ) = d(f_1^\circ \cap f_2^\circ) \wedge (f_1 \cap f_2) \neq f_1 \wedge (f_1 \cap f_2) \neq f_2$	2/2	$T * T - * * * - T * T$
		1/1	$1 * T - * * * - T * T$
`cover (v)`	$(f_2 \cap f_1 = f_2) \wedge (f_2^\circ \cap f_1^\circ) \neq \emptyset) \wedge (f_1 - f_1^\circ) \cap (f_2 - f_2^\circ) \neq \emptyset$	2/2, 2/1, 1/1	$T * T - * TT - FFT$
`coveredby (b)`	$(f_1 \cap f_2 = f_1) \wedge (f_1^\circ \cap f_2^\circ) \neq \emptyset) \wedge (f_1 - f_1^\circ) \cap (f_2 - f_2^\circ) \neq \emptyset$	2/2, 1/1, 1/2	$T * F - * TF - TTT$

Table 3.1: Definition of the reference set of topological relations.

- $\mathcal{T}_d$ the set $\{\theta_{d_1,d_2} | \theta \in \mathcal{T}, d_1, d_2 \in DIM,\ \theta$ is defined for $d_1, d_2\}$;
- $\mathcal{T}_{d_1,d_2}$ the set $\{\theta_{d_1,d_2} | d_1, d_2 \in DIM,\ \theta_{d_1,d_2} \in \mathcal{T}_d\}$;
- $I_9(\theta_{d_1,d_2})$ the set of the 9-intersection matrices associated with θ_{d_1,d_2};
- $\mathcal{I}_9$ the set $\{ I_9(\theta_{d_1,d_2}) | \theta_{d_1,d_2} \in \mathcal{T}_d \}$of all 9-intersection matrices.

3.2.2 A Similarity Function for Consistency Checking

The topological similarity function we propose here has been developed with the particular purpose of comparing distinct datasets referring the same or overlapping areas, possibly representing the same features with different dimensions, in order to evaluate the opportunity of using them together. More precisely, such function can be used for: (i) consistency analysis of two datasets in order to evaluate if they may lead to contradictory results during query processing; (ii) similarity analysis of two datasets in order to detect if they represent the same or overlapping area in a similar way, before starting a more accurate comparison of their content. The rationale behind such function is to capture the "main" differences

between two existing datasets through a coarse analysis of the topological relations and dimensions among the features they have in common.

In order to define a similarity function for topological relationships in $\mathcal{T}$, since each topological relationship in $\mathcal{T}$ corresponds to a set of 9-intersection matrices, we use a two steps approach: first a distance function between two 9-intersection matrices is defined, then such function is used in computing the final result.

In defining the similarity between two 9-intersection matrices ψ_1 and ψ_2 (denoted by $d_9(\psi_1, \psi_2)$), we adopt the approach proposed in [EAT92] and we define it as the fraction between the number of different cells in the two matrices and the total number of cells (9). Two cells are considered different if one corresponds to a non-empty intersection (whatever is its dimension) and the other to an empty intersection. Note that the dimension of the intersection is not taken into account to compare two matrices, it is considered only to discriminate between `overlap` and `cross` (as we can see from Table 3.1). Moreover,it is important to highlight that in this function each cell has the same weight 1, since in this case our purpose is to simply compare two different 9-intersection matrices.

Based on this distance, given two relationships θ_1 and θ_2 in $\mathcal{T}$, their similarity can now be computed subtracting from 1 (maximum similarity) the minimum distance between any 9-intersection matrix defining θ_1 and any 9-intersection matrix defining θ_2, that represents the differences between the two relationships. We have chosen the minimum and not the average or other functions since we are interested in maximizing similarity among topological relations after dimension changes.

Definition 3 (*Topological similarity*) Let $d_1, d_2, d_3, d_4 \in DIM$ and let $\theta^1_{d_1,d_2}$, $\theta^2_{d_3,d_4} \in \mathcal{T}_d$. The topological similarity between $\theta^1_{d_1,d_2}$ and $\theta^2_{d_3,d_4}$ is defined as follows: $ts : \mathcal{T}_d \times \mathcal{T}_d \rightarrow [0, 1]$, where:

$$ts(\theta^1_{d_1,d_2}, \theta^2_{d_3,d_4}) = 1 - min\{d_9(\psi_1, \psi_2) | \psi_1 \in I_9(\theta^1_{d_1,d_2}), \psi_2 \in I_9(\theta^2_{d_3,d_4})\}. \ \square$$

Notice that, for relations θ_{d_1,d_2} defined over points (i.e., those with $d_1 = 0$ or $d_2 = 0$), since points have no boundary, the topological similarity is computed by excluding from the computation of d_9 the matrix cells corresponding to boundary.

Example 7 *Suppose to compute* $ts(\mathtt{c}_{2,1}, \mathtt{i}_{1,2})$. *Based on Table 3.1, it is possible to show that* $ttc_{2,1}$ *corresponds to the following two 9-intersection matrices:*

$$\mathtt{c}^1_{2,1} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix} \mathtt{c}^2_{2,1} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

On the other hand, $\mathtt{i}_{1,2}$ *corresponds to the following two 9-intersection matrices:*

$$\mathtt{i}^1_{1,2} = \begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix} \mathtt{i}^2_{1,2} = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$$

In order to compute their distance, we first have to compute all the distances between a matrix for `contain` *and a matrix for* `in`*, i.e.,* $d_9(\mathtt{c}^{\mathtt{i}}_{2,1}, \mathtt{i}^{\mathtt{j}}_{1,2})$*,* $i, j = 1, 2$*, and then take the minimum value. For example,* $d_9(\mathtt{c}^1_{2,1}, \mathtt{i}^1_{1,2}) = 6/9$*, since 6 positions out of 9 are different. Similarly,* $d_9(\mathtt{c}^1_{2,1}, \mathtt{i}^2_{1,2}) = 5/9$*,* $d_9(\mathtt{c}^2_{2,1}, \mathtt{i}^1_{1,2}) = 5/9$*,* $d_9(\mathtt{c}^2_{2,1}, \mathtt{i}^2_{1,2}) = 4/9$*. Thus,* $ts(\mathtt{c}_{2,1}, \mathtt{i}_{1,2}) = 1 - min\{4/9, 6/9, 5/9\} = 1 - 4/9 = 6/9$*.* □

We notice that the similarity function presented in Definition 3 takes into account the dimension of the features over which topological relations are applied in order to compute their similarity. Indeed, different pairs of dimensions may lead to different sets of 9-intersection matrices representing the same topological relation. This is more evident from the patterns presented in Table 3.1 where, for example, for relation *touch*, different patterns are associated with different pairs of object dimensions.

Table 3.2 and Table 3.3 presents the values for $ts(\theta^1_{d_1,d_2}, \theta^2_{d_3,d_4})$, considering all dimensions and all topological relations. Since, as we explained in Proposition 3 of Section 3.4, the topological similarity function is simmetric, in Tables 3.2 and 3.3 the similarity values are presented only once for each pairs of dimensions (e.g., the value for the pairs of dimensions (1,2) (2,1) is the same as that between (2,1) and (1,2)). We also notice that, for pairs based on the same dimensions, (e.g., the pairs (1,2) (1,2)) for each topological relation the most similar one is the relation itself, thus the value 1 appears only on the diagonal. For pairs based on different dimensions, it is useful to discriminate between two situations depending on the presence of dimension 0 in the pairs. If dim 0 does not appear, the highest similarity value is associated with cells (θ,θ), i.e. to pairs of identical topological relations (e.g., considering the pairs (2,1), (1,2), the similarity value 1 is associated with cells (d,d), (t,t), (r,r)). If dim 0 appears this is not true, due to boundary information. , that is not present when considering points.

3.2.3 A Similarity Function For Relaxed Query Processing

In this section we present a similarity function to be used in relaxed query processing to improve the quality, in terms of completeness and accuracy, of the obtained result in case of missing data or non-well-specified queries.

Similarity values computed by the function presented in Section 3.2.2 just depends on the number of corresponding cells which are equal in the considered sets of 9-intersection

		(2,2)								(2,1)					(2,0) (1,0)			(1,2)				
		d	t	i	b	e	c	v	o	d	t	c	v	r	d	t	c	d	t	i	b	r
(2,2)	d	1	0.88	0.55	0.44	0.33	0.55	0.44	0.55	1	0.88	0.55	0.55	0.77	1	0.77	0.77	1	0.88	0.55	0.55	0.77
	t	0.88	1	0.44	0.55	0.44	0.44	0.55	0.66	0.88	1	0.44	0.66	0.77	1	0.77	0.77	0.88	1	0.44	0.66	0.77
	i	0.55	0.44	1	0.88	0.55	0.33	0.22	0.55	0.55	0.66	0.44	0.44	0.77	0.55	0.55	0.55	0.55	0.55	1	0.88	0.77
	b	0.44	0.55	0.88	1	0.66	0.22	0.33	0.66	0.44	0.66	0.33	0.55	0.77	0.55	0.55	0.55	0.44	0.66	0.88	1	0.77
	e	0.33	0.44	0.55	0.66	1	0.55	0.66	0.33	0.33	0.55	0.55	0.77	0.55	0.55	0.55	0.77	0.33	0.55	0.55	0.77	0.55
	c	0.55	0.44	0.33	0.22	0.55	1	0.88	0.55	0.55	0.55	1	0.88	0.77	0.77	0.77	1	0.55	0.66	0.44	0.44	0.77
	v	0.44	0.55	0.22	0.33	0.66	0.88	1	0.66	0.44	0.66	0.88	1	0.77	0.77	0.77	1	0.44	0.66	0.33	0.55	0.77
	o	0.55	0.66	0.55	0.66	0.33	0.55	0.66	1	0.55	0.77	0.66	0.77	0.88	0.77	0.77	0.77	0.55	0.77	0.66	0.77	0.88
(2,1)	d									1	0.88	0.55	0.55	0.77	1	0.77	0.77	1	0.88	0.55	0.55	0.77
	t									0.88	1	0.66	0.88	0.88	1	1	0.77	0.88	1	0.66	0.66	0.77
	c									0.55	0.66	1	0.88	0.88	0.77	0.88	1	0.55	0.66	0.55	0.44	0.77
	v									0.55	0.88	0.88	1	0.88	0.77	0.88	1	0.55	0.66	0.44	0.55	0.77
	r									0.77	0.88	0.88	0.88	1	0.77	0.77	0.77	0.77	0.77	0.77	0.77	1
(2,0)	d														1	0.77	0.77	1	1	0.55	0.66	0.88
(1,0)	t														0.77	1	0.77	0.77	0.77	0.55	0.55	0.77
	c														0.77	0.77	1	0.77	0.77	0.55	0.66	0.88
(1,2)	d																	1	0.88	0.55	0.55	0.77
	t																	0.88	1	0.66	0.88	0.88
	i																	0.55	0.66	1	0.88	0.88
	b																	0.55	0.88	0.88	1	0.88
	r																	0.77	0.88	0.88	0.88	1

Table 3.2: Similarity values for topological relationships

matrices. This approach usually leads to two problems: (i) the same similarity value is often assigned to different pairs of topological relations, since all pairs of corresponding cells with the same content are treated in the same way; (ii) it may happen that some topological relations have `disjoint` as the most similar relation. We believe that a good similarity function, to be used for relaxation purposes, should overcome such drawbacks in order to obtain a progressive relaxation and to avoid the relaxation of a given relation with `disjoint`. In particular, we claim that the following properties should be satisfied by an"ideal" similarity function suitable for relaxation purposes:

- **P1**: Given a topological relation r, there do not exist two relations $r1$ and $r2$ such that $ts(r, r1) = ts(r, r2)$.
- **P2**: Given a relation r, except `disjoint`, in the total order induced by similarity values between r and all the other topological relations, the highest relation is r, the lowest is `disjoint`.

Our purpose is not to define the "ideal" function; rather, we believe that in real cases "almost-ideal" functions are often adequate. In order to take into account property **P1**, we propose to compare two 9-intersection matrices ψ_1 and ψ_2 using a specific weight matrix W, thus weighting the comparison of different pairs of cells and therefore better discriminating between different configurations. Additionally in order to compare the cells between the two matrices, we map the value $\oslash$ in 0 and the value $\neq \oslash$ in 1.

Definition 4 (*Weighted 9-intersection matrix similarity function*) Let ψ_1 and ψ_2 be two 9-intersection matrices and let $W = \{w_{i,j} | i, j = 1, 2, 3\}$ be a weights matrix. The weighted

		(1,1)									(0,1)		(0,2)	(0,0)	
		d	t	i	b	e	c	v	o	r	d	t	i	d	e
(2,2)	d	1	0.88	0.55	0.44	0.33	0.55	0.44	0.88	0.88	1	0.77	0.77	1	0.66
	t	0.88	1	0.66	0.66	0.55	0.55	0.66	0.88	0.88	1	0.77	0.77	1	0.66
	i	0.55	0.77	1	0.88	0.55	0.33	0.22	0.88	0.88	0.77	0.77	1	0.77	0.88
	b	0.44	0.77	0.88	1	0.66	0.22	0.33	0.88	0.88	0.77	0.77	1	0.77	0.88
	e	0.33	0.66	0.55	0.66	1	0.55	0.66	0.77	0.77	0.55	0.55	0.77	0.66	1
	c	0.55	0.77	0.33	0.22	0.55	1	0.88	0.88	0.88	0.55	0.55	0.55	0.77	0.88
	v	0.44	0.77	0.22	0.33	0.66	0.88	1	0.88	0.88	0.55	0.55	0.55	0.77	0.88
	o	0.55	0.77	0.55	0.66	0.33	0.55	0.66	0.88	0.88	0.77	0.77	0.77	0.88	0.77
(2,1)	d	1	0.88	0.55	0.44	0.33	0.55	0.44	0.88	0.88	1	0.77	0.77	1	0.66
	t	0.88	1	0.66	0.66	0.55	0.55	0.66	0.88	0.88	1	0.77	0.77	1	0.77
	c	0.55	0.77	0.44	0.33	0.55	1	0.88	0.88	0.88	0.55	0.55	0.55	0.77	0.88
	v	0.55	0.77	0.44	0.55	0.77	0.88	1	0.88	0.88	0.66	0.55	0.66	0.77	0.88
	r	0.77	0.88	0.77	0.77	0.55	0.77	0.77	1	1	0.88	0.77	0.88	0.88	0.77
(2,0)	d	1	1	0.55	0.55	0.55	0.77	0.77	0.88	0.88	1	0.88	0.77	1	0.66
(1,0)	t	0.77	0.88	0.55	0.55	0.55	0.77	0.77	0.77	0.77	0.88	0.77	0.66	0.88	0.77
	c	0.77	0.77	0.55	0.55	0.77	1	1	0.88	0.88	0.77	0.66	0.77	0.77	0.88
(1,2)	d	1	0.88	0.55	0.44	0.33	0.55	0.44	0.88	0.88	1	0.77	0.77	1	0.66
	t	0.88	1	0.55	0.66	0.55	0.66	0.66	0.88	0.88	1	1	0.77	1	0.77
	i	0.55	0.77	1	0.88	0.55	0.44	0.33	0.88	0.88	0.77	0.88	1	0.77	0.88
	b	0.55	0.77	0.88	1	0.77	0.44	0.55	0.88	0.88	0.77	0.88	1	0.77	0.88
	r	0.77	0.88	0.77	0.77	0.55	0.77	0.77	1	1	0.77	0.77	0.77	0.88	0.77
(1,1)	d	1	0.88	0.55	0.44	0.33	0.55	0.44	0.88	0.88	1	0.77	0.77	1	0.66
	t	0.88	1	0.77	0.77	0.66	0.77	0.77	0.88	0.88	1	0.88	0.77	1	0.66
	i	0.55	0.77	1	0.88	0.55	0.33	0.22	0.88	0.88	0.77	0.77	1	0.77	0.88
	b	0.44	0.77	0.88	1	0.66	0.22	0.33	0.88	0.88	0.77	0.77	1	0.77	0.88
	e	0.33	0.66	0.55	0.66	1	0.55	0.66	0.77	0.77	0.55	0.55	0.77	0.66	1
	c	0.55	0.77	0.33	0.22	0.55	1	0.88	0.88	0.88	0.55	0.55	0.55	0.77	0.88
	v	0.44	0.77	0.22	0.33	0.66	0.22	1	0.88	0.88	0.55	0.55	0.55	0.77	0.88
	o	0.88	0.88	0.88	0.88	0.77	0.88	0.44	1	1	0.88	0.77	0.88	0.88	0.77
	r	0.88	0.88	0.88	0.88	0.77	0.88	0.44	1	1	0.88	0.77	0.88	0.88	0.77
(0,2)	d										1	0.77	0.77	1	0.66
(0,1)	t										0.77	1	0.77	0.88	0.77
	i										0.77	0.77	1	0.77	0.88
(0,0)	d													1	0.66
	e													0.66	1

Table 3.3: Similarity values for topological relationships.

similarity between ψ_1 and ψ_2, denoted by $ws_9(\psi_1, \psi_2)$, is defined as follows: $ws_9 : \mathcal{I}_9 \times \mathcal{I}_9 \rightarrow [0, 1]$, where:

$$ws_9(\psi_1, \psi_2) = \frac{\sum_{i,j}(w_{i,j} * (1 - \mid \psi_1[i,j] - \psi_2[i,j] \mid))}{\sum_{i,j} w_{i,j}}$$

□

In order to define the weights matrix, we consider the following heuristic: *intersections involving interiors are more relevant in defining similarity than those involving boundaries, which are more relevant than those involving exteriors*. To this aim, the weight of the interior-interior position is set to 54 which is 1 plus the sum of all the other weights in the matrix. This reasoning is applied to all the matrix-cells, resulting in W = ((54 18 3), (18 9 1), (3 1 0)). Other matrices can of course be taken into account without compromising the correctness of the relaxing approach.

Example 8 *Consider matrices* $\mathsf{c}^1_{2,1}$, $\mathsf{c}^2_{2,1}$ *presented in Example 7. By using* W, *the previous introduced weights matrices* ws_9 *is computed as follows:* $ws_9(\mathsf{c}^1_{2,1}, \mathsf{c}^2_{2,1})$*=(54*(1-(1-1))+18*(1-(1-1))+3*(1-(1-1))+18*(1-(0-1))+*

9(1-(0-0))+1*(1-(1-1)) +3*(1-(0-0))+1*(1-(0-0))+0)/107 = (54+18+3+0+9+1+3+1) / 107 = 0.831.* □

In order to take into account property **P2**, since in matrices that represent `disjoint` all cells in *inner* positions (1,1 - 1,2 - 2,1 - 2,2) are empty (while the others are not), we introduce two additional indexes between two 9-intersection matrices which count the number of corresponding cells in specific positions: (i) an *inner index* ii_4, counting the number of common not empty cells in inner positions, representing intersections between interiors and boundaries; (ii) an *outer index* oi_5, counting the number of common not empty cells in other positions, representing intersections among objects and exteriors. Both indexes are then normalized between 0 and 1.

The topological similarity function can now be defined as follows: (i) for each pair of 9-intersection matrices belonging to the representation of the two input topological relations, compute the weighted similarity, index ii_4, index oi_5, and take their average value; (ii) take the maximum value computed so far as similarity value.

Definition 5 (*Weighted topological similarity function*) Let $\theta^1_{d_1,d_2}, \theta^2_{d_3,d_4} \in \mathcal{T}_d$ with d_1, d_2, d_3, $d_4 \in DIM$. The weighted topological similarity between $\theta^1_{d_1,d_2}$ and $\theta^2_{d_3,d_4}$, denoted by $wts(\theta^1_{d_1,d_2}, \theta^2_{d_3,d_4})$, is defined as follows: $wts : \mathcal{T}_d \times \mathcal{T}_d \rightarrow [0,1]$, where:

$$wts(\theta^1_{d_1,d_2}, \theta^2_{d_3,d_4}) = max\{\frac{ws_9(\psi_1,\psi_2) + ii_4(\psi_1,\psi_2) + oi_5(\psi_1,\psi_2)}{3} \mid \psi_1 \in I_9(\theta^1_{d_1,d_2}), \psi_2 \in I_9(\theta^2_{d_3,d_4})\}.$$

□

Table 3.4 and Table 3.5 present all the values of the weighted topological similarity for all the relations defined for the dimensions region, line and point. The proposed similarity function is quite suitable for a relaxed processing. Indeed, while it compares topological relations based on 9-intersection matrices, `disjoint` is never the most similar topological relation with respect to all other topological relations (with the exception of `disjoint` and `touch`).

The high similarity between `touch` and `disjoint` cannot be avoided without considering proximity measures, but the usage and the analysis of distance information is out of the scope of the activities of this thesis. Moreover, such function performs a good discrimination, thus each topological relations is gradually relaxed into an ordered sequence of sets of relations.

		(2,2)								(2,1)					(2,0) (1,0)			(1,2)				
		d	t	i	b	e	c	v	o	d	t	c	v	r	d	t	c	d	t	i	b	r
(2,2)	d	1	0.889	0.463	0.352	0.345	0.463	0.352	0.358	1	0.889	0.463	0.491	0.609	1	0.661	0.509	1	0.889	0.463	0.491	0.609
	t	0.889	1	0.519	0.463	0.457	0.519	0.463	0.470	0.889	1	0.519	0.603	0.664	1	0.661	0.509	0.889	1	0.519	0.603	0.664
	i	0.463	0.519	1	0.889	0.770	0.863	0.752	0.603	0.463	0.779	0.863	0.891	0.854	0.179	0.493	0.645	0.463	0.692	1	0.916	0.785
	b	0.352	0.463	0.889	1	0.715	0.752	0.863	0.715	0.352	0.724	0.891	0.975	0.918	0.179	0.493	0.645	0.352	0.636	0.916	1	0.840
	e	0.345	0.457	0.770	0.715	1	0.770	0.715	0.430	0.345	0.630	0.770	0.854	0.639	0.326	0.513	0.816	0.345	0.630	0.770	0.854	0.639
	c	0.463	0.519	0.863	0.752	0.770	1	0.889	0.603	0.463	0.692	1	0.916	0.785	0.509	0.696	1	0.463	0.779	0.863	0.891	0.854
	v	0.352	0.463	0.752	0.863	0.715	0.889	1	0.715	0.352	0.636	0.916	1	0.840	0.509	0.696	1	0.352	0.724	0.891	0.975	0.918
	o	0.358	0.470	0.603	0.715	0.430	0.603	0.715	1	0.358	0.609	0.743	0.854	0.930	0.363	0.509	0.661	0.358	0.609	0.743	0.854	0.930
(2,1)	d									1	0.889	0.463	0.491	0.609	1	0.661	0.509	1	0.889	0.463	0.491	0.609
	t									0.889	1	0.692	0.776	0.804	1	1	0.696	0.889	1	0.779	0.751	0.692
	c									0.463	0.692	1	0.916	0.924	0.509	0.696	1	0.463	0.779	0.975	0.891	0.918
	v									0.491	0.776	0.916	1	0.924	0.509	0.696	1	0.491	0.751	0.891	0.975	0.918
	r									0.609	0.804	0.924	0.924	1	0.363	0.509	0.661	0.609	0.692	0.918	0.918	1
(2,0)	d														1	0.661	0.509	1	1	0.179	0.422	0.605
(1,0)	t														0.661	1	0.696	0.661	0.741	0.493	0.583	0.679
	c														0.509	0.696	1	0.509	0.589	0.645	0.887	0.983
(1,2)	d																	1	0.889	0.463	0.491	0.609
	t																	0.889	1	0.692	0.776	0.804
	i																	0.463	0.692	1	0.916	0.924
	b																	0.491	0.776	0.916	1	0.924
	r																	0.609	0.804	0.924	0.924	1

Table 3.4: Similarity values for topological relations.

3.2.4 Comparison Between the Two Similarity Functions

In this section we briefly compare the two similarity functions presented in Section 3.2.2 , denoted by f_{cc}, and in Section 3.2.3, denoted by f_{rp} considering their main analogies and differences.

As pointed out in Section 3.2.3 function f_{rp} has been created to overcome the following problems of function f_{cc}: (PB1) the same similarity value is often assigned to different pairs of topological relations and (PB2) it may happen that some topological relations have `disjoint` as the most similar relation.

Thus, to one side, the "“"'good"”"' properties already discussed for function f_{cc}, such as the position of similarity value 1 in the tables, are satisfied by also by function f_{rp}, since it is the behaviour we desire from a a similarity function.

On the other side, the main differences are related to the problems of function f_{cc} and in particular: (i) function f_{cc} has only 7 different values of similarity, while function f_{qp} has 66 different values of similarity. This is an important characteristic that we have introduced in the second function in order to better discriminate the several degrees of similarity among the topological relations (PB1).

The increasing variety of similarity value considered have two consequences: first of all *disjoint* has always the lowest similarity value, except for `disjoint` and `touch` (PB2).

Moreover, given two pairs of dimensions and one topological relation defined for the first pair of dimension, supposing to classify the topological relations of the second dimensions in decreasing order of similarity, we obtain two different classification for the two similarity function. This is due to the different issues the two functions address.

		(1,1)									(0,1)		(0,2)	(0,0)	
		d	t	i	b	e	c	v	o	r	d	t	i	d	e
(2,2)	d	1	0.889	0.463	0.352	0.345	0.463	0.352	0.748	0.748	1	0.661	0.509	1	0.333
	t	0.889	1	0.519	0.463	0.457	0.519	0.463	0.804	0.804	1	0.661	0.509	1	0.333
	i	0.463	0.763	1	0.889	0.770	0.863	0.752	0.924	0.924	0.509	0.696	1	0.350	0.983
	b	0.352	0.763	0.889	1	0.715	0.752	0.863	0.924	0.924	0.509	0.696	1	0.350	0.983
	e	0.345	0.596	0.770	0.715	1	0.770	0.715	0.848	0.848	0.326	0.513	0.816	0.333	1
	c	0.463	0.763	0.863	0.752	0.770	1	0.889	0.924	0.924	0.179	0.493	0.645	0.350	0.983
	v	0.352	0.763	0.752	0.863	0.715	0.889	1	0.924	0.924	0.179	0.493	0.645	0.350	0.983
	o	0.358	0.637	0.603	0.715	0.430	0.603	0.715	0.889	0.889	0.363	0.509	0.661	0.367	0.967
(2,1)	d	1	0.889	0.463	0.352	0.345	0.463	0.352	0.748	0.748	1	0.661	0.509	1	0.333
	t	0.889	1	0.779	0.724	0.630	0.692	0.636	0.804	0.804	1	0.741	0.589	1	0.350
	c	0.463	0.791	0.863	0.891	0.770	1	0.916	0.988	0.988	0.179	0.493	0.645	0.350	0.983
	v	0.491	0.763	0.891	0.975	0.854	0.916	1	0.988	0.988	0.422	0.583	0.887	0.350	0.983
	r	0.609	0.776	0.854	0.918	0.639	0.785	0.840	1	1	0.605	0.679	0.983	0.367	0.967
(2,0) (1,0)	d	1	1	0.179	0.179	0.326	0.509	0.509	0.605	0.605	1	0.983	0.350	1	0.333
	t	0.661	0.983	0.493	0.493	0.513	0.696	0.696	0.679	0.679	0.983	0.967	0.333	0.983	0.350
	c	0.509	0.679	0.645	0.645	0.816	1	1	0.983	0.983	0.350	0.333	0.967	0.350	0.983
(1,2)	d	1	0.889	0.463	0.352	0.345	0.463	0.352	0.748	0.748	1	0.661	0.509	1	0.333
	t	0.889	1	0.692	0.636	0.630	0.779	0.724	0.804	0.804	1	1	0.696	1	0.350
	i	0.463	0.791	1	0.916	0.770	0.863	0.891	0.988	0.988	0.509	0.696	1	0.350	0.983
	b	0.491	0.763	0.916	1	0.854	0.891	0.975	0.988	0.988	0.509	0.696	1	0.350	0.983
	r	0.609	0.776	0.785	0.840	0.639	0.854	0.918	1	1	0.363	0.509	0.661	0.367	0.967
(1,1)	d	1	0.889	0.463	0.352	0.345	0.463	0.352	0.748	0.748	1	0.661	0.509	1	0.333
	t	0.889	1	0.763	0.763	0.596	0.763	0.763	0.804	0.804	1	0.983	0.679	1	0.333
	i	0.463	0.763	1	0.889	0.770	0.863	0.752	0.924	0.924	0.509	0.696	1	0.350	0.983
	b	0.352	0.763	0.889	1	0.715	0.752	0.863	0.924	0.924	0.509	0.696	1	0.350	0.983
	e	0.345	0.596	0.770	0.715	1	0.770	0.715	0.848	0.848	0.326	0.513	0.816	0.333	1
	c	0.463	0.763	0.863	0.752	0.770	1	0.889	0.924	0.924	0.179	0.493	0.645	0.350	0.983
	v	0.352	0.763	0.752	0.863	0.715	0.889	1	0.924	0.924	0.179	0.493	0.645	0.350	0.983
	o	0.748	0.804	0.924	0.924	0.848	0.924	0.924	1	1	0.605	0.679	0.983	0.367	0.967
	r	0.748	0.804	0.924	0.924	0.848	0.924	0.924	1	1	0.605	0.679	0.983	0.367	0.967
(0,2) (0,1)	d										1	0.661	0.509	1	0.333
	t										0.661	1	0.696	0.983	0.350
	i										0.509	0.696	1	0.350	0.983
(0,0)	d													1	0.333
	e													0.333	1

Table 3.5: Similarity values for topological relations.

3.3 Direction Relationships

In the following, we first present a formal model for direction relations; then, a similarity function for comparing two direction relationships, possibly defined over multi-resolution pairs of features, with the aim of consistency checking is provided.

3.3.1 The Model

In defining our reference model for direction relations, we rely on the 5×5 matrix model (denote also with 25-intersections model) proposed in [GE00, Goy00] and then formalized in [SK04, SK02] for connected and disconnected regions (see Chapter 2 for additional details). Here, we extend this model to deal with regions, lines, and points for reference and target objects. For the sake of simplicity, we however consider only connected objects.

Depending on the dimension and shape of the reference object, a different number of tiles is generated. Figure 3.2 presents tiles generated when the reference object is a region (REG) (or a sideway line, S_LINE), a vertical line (V_LINE), a horizontal line (H_LINE), or a point (P). We denote with $ROBJ = REG \cup S\_LINE \cup V\_LINE \cup H\_LINE \cup H\_LINE \cup P$.

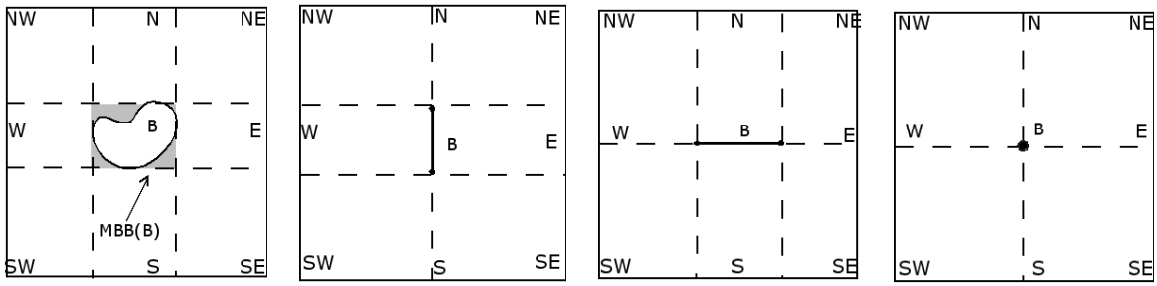


Figure 3.2: Different grids according to different reference object dimension: (a) region; (b) vertical line (c) horizontal line (d) point.

Since the 5×5 matrix model leads to a huge number of different configurations, following what has been previously done in [GE00, Goy00, SK04, SK02], we group direction relations into sets. The group of relationships we consider in this chapter corresponds to the set of traditional direction relationships ($\mathcal{C}$={`northwest (nw), north (n), northeast (ne), west (w),`
`minimum bounding box (mbb), east (e), southeast (se), south (s), soutwest (sw)`}), possibly combined together.

We remark that other groups of matrices can be defined. For example, a relation for each portion of space considered in the 5×5 matrix can also be considered. We believe however that, from a user point of view, it is more convenient to cope with a small and well known set of relationships, as already discussed for topological relations (see Section 3.2). This choice is also motivated by the fact that, as far as we know, no commercial system directly support direction relations.

Similarly to topological relations, Table 3.6 and 3.7 presents the patterns for single-tile relations. In particular, the first column presents the direction relations and the corresponding 5×5 matrix, in the second and third column are presented the dimension of the reference and target object, respectively. Finally, the two last columns present the pattern and the constraints on the pattern, defining the configurations grouped under the direction relation pointed out in the first column. In this case, the pattern is a string $P_{1,1}P_{1,2}P_{1,3}P_{1,4}P_{1,5}$ $P_{2,1}P_{2,2}P_{2,3}P_{2,4}P_{2,5}$ $P_{3,1}P_{3,2}P_{3,3}P_{3,4}P_{3,5}$ $P_{4,1}P_{4,2}P_{4,3}$ $P_{4,4}P_{4,5}$ $P_{5,1}P_{5,2}P_{5,3}P_{5,4}P_{5,5}$ where element $P_{i,j}$ corresponds to cell (i, j) in the 5×5 matrix. The symbol $*$ means that the corresponding position is not relevant, F and T mean that the intersection is or is not empty.

Notice that, in the case of non-connected obejcts, some constraints for points and lines in Tables 3.6 and 3.7, for relations `n, e, s, w`, are no longer satisfied. Take for example relation n; the constraint ($P_{1,2}$=T $\wedge$ $P_{1,4}$=T) $\Rightarrow$ $P_{1,3}$=T for lines cannot be satisfied if we

Name	Refer. Obj. type B	Target Obj. type A	Pattern	Constraints
nw $\begin{bmatrix} P_{1,1} & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \end{bmatrix}$	$ROBJ$	OBJ	$P_{1,1}$=T	
n $\begin{bmatrix} F & P_{1,2} & P_{1,3} & P_{1,4} & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \end{bmatrix}$	$REG \cup S\_LINE \cup H\_LINE$	REG	$P_{1,3}$=T $P_{1,4}$=∗ $P_{1,2}$=∗	
		$LINE$	$\exists\ j \in \{2,3,4\}$. $P_{1,j}$ = T ∧ (($P_{1,2}$=T ∧ $P_{1,4}$=T) ⇒ $P_{1,3}$=T)	
		$POINT$	$\exists! j \in \{2,3,4\}.P_{1,j}$=T	
	$V\_LINE \cup POINT$	REG	not defined	
		$LINE \cup POINT$	$P_{1,2}$=F $P_{1,3}$=T $P_{1,4}$=F	
ne $\begin{bmatrix} F & F & F & F & P_{1,5} \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \end{bmatrix}$	$ROBJ$	OBJ	$P_{1,5}$=T	
w $\begin{bmatrix} F & F & F & F & F \\ P_{2,1} & F & F & F & F \\ P_{3,1} & F & F & F & F \\ P_{4,1} & F & F & F & F \\ F & F & F & F & F \end{bmatrix}$	$REG \cup S\_LINE \cup V\_LINE$	REG	$P_{3,1}$=T $P_{2,1}$=∗ $P_{4,1}$=∗	
		$LINE$	$\exists i \in \{2,3,4\}.P_{i,1}$=T ∧ (($P_{2,1}$=T ∧ $P_{4,1}$=T)⇒ $P_{3,1}$=T)	
		$POINT$	$\exists! i \in \{2,3,4\}.P_{i,1}$=T	
	$H\_LINE \cup POINT$	REG	not defined	
		$LINE \cup POINT$	$P_{3,1}$=T $P_{2,1}$=F, $P_{4,1}$=F	

Table 3.6: Patterns associated with single-tile relations.

consider a non-connected line object, made by two distinct segments, one positioned in $P_{1,2}$ and positioned in $P_{1,4}$. Indeed, in this case $P_{1,2}$=$P_{1,4}$=T does not imply $P_{1,3}$=T. Similarly, the constraint $\exists! j \in \{2,3,4\}.P_{1,j}$=T for points does not hold for non-connected objects. In fact, if we consider an object made by two distinct points, positioned in $P_{1,2}$ and $P_{1,4}$, we have $P_{1,2}$=$P_{1,4}$=T, thus the true value is not unique as required by the constraint.

In order to make direction relations pairwise disjoint, it is necessary to decide how to manage the boundary between two adjacent different tiles, for example the line between N and NE, in order to be able to assign a point or a line that lies on it to only one between N and NE. We decide to assign the boundary between to adjacent tiles only to the direction relations among {`n, w, e, s`} (the most relevant ones). Additionally, we assume that `mbb` contains the boundary of the corresponding tile. As a consequence, `nw, ne, se, sw` are open regions while `mbb` is a closed region.

The overall set of direction relationships can now be defined as follows.

Definition 6 (*Set of Direction Relationships*) The set of direction relationships, called $\mathcal{C}$, is composed of any expression χ of the form $R_1 : \ldots : R_k$, 1≤ k ≤ 9, such that:

(i) $R_i \in \{\texttt{mbb}, \texttt{s}, \texttt{sw}, \texttt{w}, \texttt{nw}, \texttt{n}, \texttt{ne}, \texttt{e}, \texttt{se}\}$, $i = 1, ..., k$;

Name	Refer. Obj. type	Target Obj. type	Pattern	Constraints
mbb $\begin{bmatrix} F & F & F & F & F \\ F & P_{2,2} & P_{2,3} & P_{2,4} & F \\ F & P_{3,2} & P_{3,3} & P_{3,4} & F \\ F & P_{4,2} & P_{4,3} & P_{4,4} & F \\ F & F & F & F & F \end{bmatrix}$	$REG \cup S\_LINE$	OBJ	$P_{3,3}$=T	$\forall P_{i,j} = T$ *with* $i,j \in \{2,3,4\}$ $\exists P_{n,m} = T$ *with* $n,m \in \{2,3,4\}$ *such that* $P_{i,j}$ *and* $P_{n,m}$ *are* 8 − *adjacent.*
	H_LINE	REG	not defined	
		$LINE$	$P_{3,3}$=T, $P_{3,2}$=∗, $P_{3,4}$=∗, $P_{i,j}$=F j∈{2,3,4}, i ∈ {2,4}	
		$POINT$	∃! j∈{2,3,4}. $P_{3,j}$=T , $P_{i,j}$=F j∈{2,3,4}, i ∈ {2,4}	
	V_LINE	REG	not defined	
		$LINE$	$P_{3,3}$=T, $P_{2,3}$=∗, $P_{4,3}$=∗, $P_{i,j}$=F i∈{2,3,4}, j∈ {2,4}	
		$POINT$	∃! i∈{2,3,4}. $P_{3,j}$=T , $P_{i,j}$=F j∈{2,3,4}, i ∈ {2,4}	
	$POINT$	$REG \cup LINE$	not defined	
		$POINT$	$P_{3,3}$=T , $P_{i,j}$=F i,j∈{1,2,3,4,5}, (i,j)≠(3,3)	
e $\begin{bmatrix} F & F & F & F & F \\ F & F & F & F & P_{2,5} \\ F & F & F & F & P_{3,5} \\ F & F & F & F & P_{4,5} \\ F & F & F & F & F \end{bmatrix}$	$REG \cup S\_LINE \cup V\_LINE$	REG	$P_{3,5}$=T $P_{2,5}$=∗ $P_{4,5}$=∗	
		$LINE$	$\exists i \in \{2,3,4\}.P_{i,5}$=T ∧ (($P_{2,5}$=T ∧ $P_{4,5}$=T) ⇒ $P_{3,5}$=T)	
		$POINT$		∃! i ∈ {2,3,4}. $P_{i,5}$=T
	$H\_LINE \cup POINT$	REG	not defined	
		$LINE \cup POINT$	$P_{3,5}$=T $P_{2,5}$=F, $P_{4,5}$=F	
sw $\begin{bmatrix} F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ P_{5,1} & F & F & F & F \end{bmatrix}$	$ROBJ$	OBJ	$P_{5,1}$=T	
s $\begin{bmatrix} F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & P_{5,2} & P_{5,3} & P_{5,4} & F \end{bmatrix}$	$REG \cup S\_LINE \cup H\_LINE$	REG	$P_{5,3}$=T $P_{5,2}$=∗ $P_{5,4}$=∗	
		$LINE$	∃ $j \in \{2,3,4\}$. $P_{5,j}$ = T ∧ (($P_{5,2}$=T ∧ $P_{5,4}$=T) ⇒ $P_{5,3}$=T)	
		$POINT$	∃!$j \in \{2,3,4\}$.$P_{5,j}$=T	
	$V\_LINE \cup POINT$	REG	not defined	
		$LINE \cup POINT$	$P_{5,2}$=F $P_{5,3}$=T $P_{5,4}$=F	
se $\begin{bmatrix} F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & F \\ F & F & F & F & P_{5,5} \end{bmatrix}$	$ROBJ$	OBJ	$P_{5,5}$=T	

Table 3.7: Patterns associated with single-tile relations.

(ii) $R_i \neq R_j$, $i \neq j$, $1 \leq i, j \leq k$;

(iii) R_i and R_{i+1} $i = 1, ..., k-1$ share at least one line boundary.

If $k = 1$, the direction relation is called *single-tile*, otherwise it is called *multi-tile*. Let $\gamma \in \mathcal{C}$, the set $\{R_1, ..., R_k\}$ is called *support* of γ and it is denoted by $s(\gamma)$. □

From the previous definition it follows that a multi-tile relation is obtained by listing a set of single-tile relations (item (i)). Item (ii) avoids the duplication of the same single-tile relation inside a multi-tile one. Finally, item (iii) provides a normal form for direction relations, specifying that two single-tile relations are adjacent in the list if and only if their corresponding tiles are adjacent in the plane. This constraint ensure that we are considering only connected relations; discarding such constraint we are able to consider also non-connected relations.

The semantics of direction relations can be defined by considering the intersection of the tiles with the target object. To this purpose, it is useful to introduce the concepts of greatest lower bound and lowest upper bound of an object on a given axis.

Definition 7 (*Upper and lower bound*) Let $A \in OBJ$, the *greatest lower bound* of the projection of A on the x-axis (respectively y-axis) is denoted by $inf_x(A)$(respectively $inf_y(A)$). The *least upper bound* of the projection of A on the x-axis (respectively y-axis) is denoted by $sup_x(A)$ (respectively $sup_y(A)$). The *minimum bounding box* of A, denoted by $mbb(A)$, is the box formed by the straight lines $x = inf_x(A)$, $x = sup_x(A)$, $y = inf_y(A)$ and $y = sup_y(A)$. □

Table 3.8 presents the semantics of single-tile direction relationships. As already presented in Chapter 2, such semantics has been obtained by extending those presented in [SK04, SK02] to deal with reference and target objects with possibly different dimensions. We notice that the proposed semantics of direction relations guarantee that single-tile expressions are mutually exclusive and represent a cover of all possible relative positions between two objects $\in OBJ$. Notice that this is not true in [SK04, SK02] (see Chapter 2 for details).

The semantics of multi-tile relationships can be defined based on the semantics of single-tile relationships as follows.

Definition 8 (*Direction Relationships*) Let $A \in OBJ$ and $B \in ROBJ$, let $\gamma \in \mathcal{C}$, $\gamma = R_1 : \ldots : R_k$. $A\ \gamma\ B$ is satisfied if and only if there exist $A_1, \ldots, A_k \in OBJ$, $A_i \neq A_j$, $i \neq j$, such that $A \supseteq A_1 \cup \ldots \cup A_k$ and relations $A_1\ R_1\ B$, $A_2\ R_2\ B$,...,$A_k\ R_k\ B$ are true. □

Name	**Definition**	**Refer. Obj. type B**
`nw`	$inf_y(A) > sup_y(B)$, $inf_x(B) > sup_x(A)$	$REG \cup S\_LINE$
	$inf_y(A) > y_B$, $inf_x(B) > sup_x(A)$	H_LINE
	$inf_y(A) > sup_y(B)$, $x_B > sup_x(A)$	V_LINE
	$inf_y(A) > y_B$, $x_B > sup_x(A)$	$POINT$
`n`	$inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$, $sup_y(B) < inf_y(A)$	$REG \cup S\_LINE$
	$y_B < inf_y(A)$, $inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$	H_LINE
	$inf_x(A) \leq x_B \leq sup_x(A)$, $inf_y(A) < sup_y(B)$	V_LINE
	$inf_x(A) \leq x_B \leq sup_x(A)$, $y_B < inf_y(A)$	$POINT$
`ne`	$inf_y(A) > sup_y(B)$, $sup_x(B) < inf_x(A)$	$REG \cup S_L INE$
	$inf_y(A) > y_B$, $sup_x(B) < inf_x(A)$	H_LINE
	$inf_y(A) > sup_y(B)$, $x_B < inf_x(A)$	V_LINE
	$inf_y(A) > y_B$, $x_B < inf_x(A)$	$POINT$
`w`	$inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$, $inf_x(B) > sup_x(A)$	$REG \cup S\_LINE$
	$sup_x(A) < inf_x(B)$, $inf_y(A) \leq y_B \leq sup_y(A)$	H_LINE
	$sup_x(A) < x_B$, $inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$	V_LINE
	$inf_y(A) \leq y_B \leq sup_y(A)$, $x_B > sup_x(A)$	$POINT$
`mbb`	$inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$, $inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$	$REG \cup S\_LINE$
	$inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$, $inf_y(A) \leq y_B \leq sup_y(A)$	H_LINE
	$inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$, $inf_x(A) \leq x_B \leq sup_x(A)$	V_LINE
	$(x_B, y_B) \in$ A	$POINT$
`e`	$inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$, $sup_x(B) < inf_x(A)$	$REG \cup S\_LINE$
	$inf_y(A) \leq y_B \leq sup_y(A)$, $sup_x(B) < inf_x(A)$	H_LINE
	$inf_y(B) \leq inf_y(A)$, $sup_y(A) \leq sup_y(B)$, $x_B < inf_x(A)$	V_LINE
	$inf_y(A) \leq y_B \leq sup_y(A)$, $x_B < inf_x(A)$	$POINT$
`sw`	$sup_y(A) < inf_y(B)$, $inf_x(B) > sup_x(A)$	$REG \cup S\_LINE$
	$sup_y(A) < y_B$, $inf_x(B) > sup_x(A)$	H_LINE
	$sup_y(A) < inf_y(B)$, $x_B > sup_x(A)$	V_LINE
	$sup_y(A) < y_B$, $x_B > sup_x(A)$	$POINT$
`s`	$inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$, $inf_y(B) > sup_y(A)$	$REG \cup S\_LINE$
	$sup_y(A) < y_B$, $inf_x(B) \leq inf_x(A)$, $sup_x(A) \leq sup_x(B)$	H_LINE
	$inf_x(A) \leq x_B \leq sup_x(A)$, $sup_y(A) < inf_y(B)$	V_LINE
	$inf_x(A) \leq x_B \leq sup_x(A)$, $sup_y(A) < y_B$	$POINT$
`se`	$sup_y(A) < inf_y(B)$, $sup_x(B) < inf_x(A)$	$REG \cup S\_LINE$
	$sup_y(A) < y_B$, $sup_x(B) < inf_x(A)$	H_LINE
	$sup_y(A) < inf_y(B)$, $x_B < inf_x(A)$	V_LINE
	$sup_y(A) < y_B$, $x_B < inf_x(A)$	$POINT$

Table 3.8: Definition of the reference set of direction relationships. A is the target object, while B is the reference one.

We remark that the previous definition does not take into account which percentage of object intersects a given tile. Rather, it assumes that, when an object intersects more tiles, it is uniformly distributed among them.

From Table 3.8, we notice that, differently from topological relationships, all single-tile relations are defined for any dimension of the reference and target object, even if the corresponding definition may change. As a consequence all multi-tile relations are defined for any dimension of the reference object and for any dimension different from point, of the target object. Indeed, it is easy to show that when the target object is a point, only single-tile relations can be defined.

In the following we adopt the following notation:

- γ_{d_1,d_2} denotes the direction relation $\gamma \in \mathcal{C}$ defined for d_1, d_2 in DIM;
- $\mathcal{C}_d$ denotes the set $\{\gamma_{d_1,d_2} \,|\gamma \in \mathcal{C}, d_1, d_2 \ \in \ DIM$, γ is defined for $d_1, d_2\}$;
- $\mathcal{C}_{d_1,d_2}$ denotes the set $\{\gamma_{d_1,d_2} \,|d_1, d_2 \in DIM, \gamma_{d_1,d_2} \in \mathcal{C}_d \}$;
- $I_{25}(\gamma_{d_1,d_2})$ denotes the set of the 25-intersection matrices associated to γ_{d_1,d_2};
- $\mathcal{I}_{25}$ the set $\{ \ I_{25}(\gamma_{d_1,d_2}) | \theta_{d_1,d_2} \in \mathcal{C}_d \ \}$ of all 25-intersection matrices.

3.3.2 A Similarity Function for Consistency Checking

Similarly to what has been done for topological relation in Section 3.2.2, the similarity function we present in the following has been designed with the aim of checking consistency and similarity of possibly multi-resolution geo-spatial maps.

In defining the similarity function for direction relations, we first notice that the approach used for topological relationships is not sufficient in this case. To show this, we first compute the difference (i.e., the distance) between two 5×5 matrices, similarly to what we have done for topological relations. Such distance, denoted by d_{25}, is defined as the fraction between the number of different cells in the two matrices and the total number of cells (25). Two cells are considered different if one corresponds to a non-empty intersection and the other to an empty intersection. Based on this distance, given two direction relationships $\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4} \in \mathcal{C}_d$, their matrix-based distance, denoted by $d_{25}(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4})$, can now be computed as the minimum distance between any 5×5 matrix defining $\gamma^1_{d_1,d_2}$ (denoted with $I_{25}(\gamma^1_{d_1,d_2})$) and any 5×5 matrix defining $\gamma^2_{d_3,d_4}$ (denoted with $I_{25}(\gamma^2_{d_3,d_4})$). The matrix-based distance just depends on the number of different single-tile relations in $\gamma^1_{d_1,d_2}$ and $\gamma^2_{d_3,d_4}$, as highlight in the following proposition.

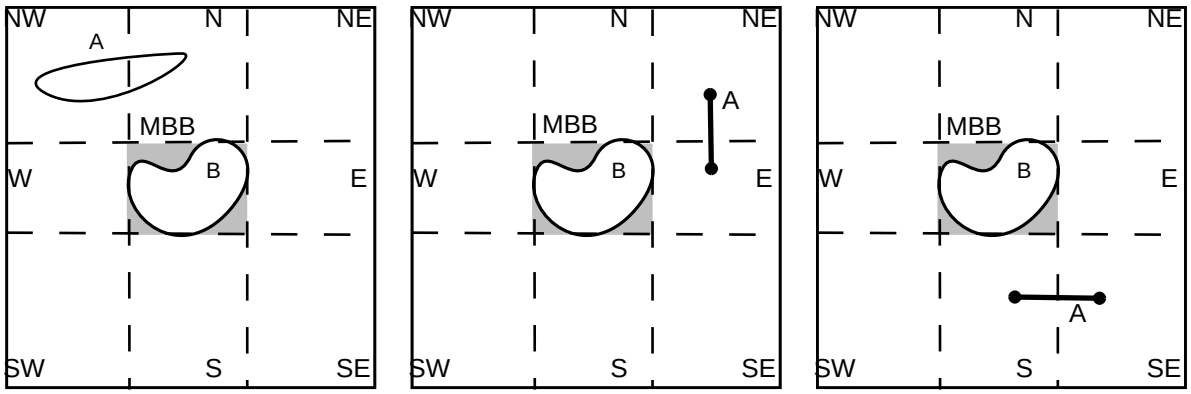


Figure 3.3: Comparing different direction relations.

Proposition 1 *Let $d_1, d_2, d_3, d_4 \in DIM$, let $\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4} \in \mathcal{C}_d$, let $s(\gamma^1_{d_1,d_2})$ the support of $\gamma^1_{d_1,d_2}$ and $s(\gamma^2_{d_3,d_4})$ the support of $\gamma^2_{d_3,d_4}$. Then, $d_{25}(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4}) = ||(s(\gamma^1_{d_1,d_2}) \cup s(\gamma^2_{d_3,d_4})) - (s(\gamma^1_{d_1,d_2}) \cap s(\gamma^2_{d_3,d_4}))||$.* □

Proof sketch. The proof is based on the fact that single-tile relations are defined for any pair of dimensions and, given two single-tile relations, the minimum number of different positions is 1. □

Based on the previous results, we can show that function d_{25} is not a good metrics for direction relations. To this end, consider the following example.

Example 9 *Consider maps M_4, M_5, M_6 shown in Figure 3.3. The target object A is a region in M_4 and becomes a line in M_5, M_6 while the reference object B is a region in all the maps. Intuitively, map M_4 seems more similar to map M_5 than to map M_6. However, computing the distance between M_4 and M_5, and then between M_4 and M_6 the result obtained is the same, i.e., $d_{25}(nw{:}n_{2,2}, ne{:}e_{1,2}) = d_{25}(nw{:}n_{2,2}, se{:}s_{1,2})$ =2/25=0.08. Thus, function d_{25} is not able to discriminate between the two situations, that are different.* □

The problem in the previous example is that function d_{25} does not take into account the distance between the two configurations in the plane. To consider this aspect, given two direction relations $\gamma^1_{d_1,d_2}$ and $\gamma^2_{d_3,d_4}$, it seems more reasonable to define a distance function which takes into account the distance in the plane among the single-tile relations contained in the support of $\gamma^1_{d_1,d_2}$ and $\gamma^2_{d_3,d_4}$. In order to define such distance, we rely on the *direction conceptual graph* (shown in Figure 2.9 in Chapter 2). The conceptual graph contains a node for each single-tile and an edge between any pair of adjacent tiles.

Definition 9 (*Minimum path*) Let γ^{1s} and $\gamma^{2s} \in \mathcal{C}$ be two single-tile relations (i.e., given two nodes of the conceptual graph), we define $Path_{Min}(\gamma^{1s}, \gamma^{2s})$ the length of the shortest path between γ^{1s} and γ^{2s} in the conceptual graph. □

The path-based distance between two single or multi-tile relations, denoted by d_{path}, is now defined by computing the average distance between each single tile of one relation and each single tile of the second one do not contained in the first one and vice versa. Function d_{path} can be defined as follows.

Definition 10 (*Path distance*) Let $\gamma^{1s}_{d_1,d_2}$,$\gamma^{2s}_{d_3,d_4} \in \mathcal{C}_d$, with d_1,d_2,d_3,$d_4 \in DIM$. Let $\gamma^1_{d_1,d_2} = R_1 : .... : R_k$ and $\gamma^2_{d_3,d_4} = S_1 : .... : S_h$. The path distance between $\gamma^{1s}_{d_1,d_2}$,$\gamma^{2s}_{d_3,d_4}$, denoted by $d_{path}(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4})$, is defined as follows:

(i) if $s(\gamma^1_{d_1,d_2}) = s(\gamma^2_{d_3,d_4})$, then 0;

(ii) if $s(\gamma^1_{d_1,d_2}) \subset s(\gamma^2_{d_3,d_4})$, then $\frac{1}{||s(\gamma^1_{d_1,d_2})|| \times ||s(\gamma^2_{d_3,d_4}) - s(\gamma^1_{d_1,d_2})||} \sum_{\substack{\gamma^{is} \in s(\gamma^1_{d_1,d_2}) \\ \gamma^{js} \in s(\gamma^2_{d_3,d_4}) - s(\gamma^1_{d_1,d_2})}} Path_{Min}(\gamma^{is}, \gamma^{js})$

(iii) if $s(\gamma^2_{d_3,d_4}) \subset s(\gamma^1_{d_1,d_2})$, then $\frac{1}{||s(\gamma^2_{d_3,d_4})|| \times ||s(\gamma^1_{d_1,d_2}) - s(\gamma^2_{d_3,d_4})||} \sum_{\substack{\gamma^{is} \in s(\gamma^2_{d_3,d_4}) \\ \gamma^{js} \in s(\gamma^1_{d_1,d_2}) - s(\gamma^2_{d_3,d_4})}} Path_{Min}(\gamma^{is}, \gamma^{js})$

(iv) in all the other cases

$\frac{1}{X+Y} \left(\sum_{\substack{\gamma^{is} \in s(\gamma^2_{d_3,d_4}) \\ \gamma^{js} \in s(\gamma^1_{d_1,d_2}) - s(\gamma^2_{d_3,d_4})}} Path_{Min}(\gamma^{is}, \gamma^{js}) + \sum_{\substack{\gamma^{is} \in s(\gamma^1_{d_1,d_2}) \\ \gamma^{js} \in s(\gamma^2_{d_3,d_4}) - s(\gamma^1_{d_1,d_2})}} Path_{Min}(\gamma^{is}, \gamma^{js}) \right)$

where $X = ||s(\gamma^1_{d_1,d_2})|| \times ||s(\gamma^2_{d_3,d_4}) - s(\gamma^1_{d_1,d_2})||$ and $Y = ||s(\gamma^2_{d_3,d_4})|| \times ||s(\gamma^1_{d_1,d_2}) - s(\gamma^2_{d_3,d_4})||$. □

Since d_{path} ranges in $[0, 4]$ value is then divide by 4 in order to get a value between 0 and 1. As shown by the following proposition, the function $Path_{Min}$ is symmetric.

Proposition 2 *Let* $\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4} \in \mathcal{C}_d$, *with* d_1,d_2,d_3,$d_4 \in DIM$*; then,* $Path_{Min}(\gamma^1_{d_1,d_2}$,$\gamma^2_{d_3,d_4})$ $= Path_{Min}(\gamma^2_{d_3,d_4}, \gamma^1_{d_1,d_2})$.

Proof sketch. Since the minimum path on a graph does not depend on the order of the starting node and the final node, the function Path_{Min} on two single-tile relations is symmetric. Considering the fact that Path_{Min} function on two multi-tile relations is based on Path_{Min} on single-tile relations, and considering the structure of the definition of Path_{Min} on multi-tile relations itself, we can conclude that it is symmetric. □.

It is easy to show that function d_{path} is symmetric and its value range between 0 and 4 (the maximum length of a path connecting two tiles). The *direction similarity* of two

direction relations $\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4}$ can now be defined as subtraction from 1 (the maximum similarity) of the difference between the two relations, computed as the product between $d_{path}(\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4})$ and their matrix based distance. The direction similarity extends the metrics presented in [GE00, Goy00] to cope with multi-tile relations.

Definition 11 (*Direction similarity*) Let $\gamma^1_{d_1,d_2}$, $\gamma^2_{d_3,d_4}$ *in* $\mathcal{C}_d$. Let $\gamma^1_{d_1,d_2} = R_1 : .... : R_k$ and $\gamma^2_{d_3,d_4} = S_1 : .... : S_h$. The direction similarity between $\gamma^1_{d_1,d_2}$ and $\gamma^2_{d_3,d_4}$, denoted by $cs(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4})$, is defined as follows:

$$cs(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4}) = 1 - d_{path}(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4}) d_{25}(\gamma^1_{d_1,d_2}, \gamma^2_{d_3,d_4}). \quad \square$$

We now give an example of the direction similarity.

Example 10 *Consider maps M_7, M_8, M_9 shown in Figure 3.4. The target object A is a region in M_7 and becomes a line in M_8 and M_9 while the reference object B is a region in all the maps.*

Intuitively, map M_7 seems more similar to map M_9 than map M_8. However, by computing the path distance between each pair of maps, we get the following values:

- *maps M_7 and M_8: $d_{path}(n_{2,2}, nw : n : ne_{1,2})$=1/4=0.25.*
- *maps M_7 and M_9: $d_{path}(n_{2,2}, nw : n_{1,2})$=1/4=0.25.*
- *maps M_8 and M_9: $d_{path}(nw : n : ne_{1,2}, nw : n_{1,2}) = \frac{3}{8}$=0.375*

Thus, the path distance alone is not sufficient to discriminate between M_7 and M_8 and M_9. However, by computing cs we get the following values:

- *maps M_7 and M_8: $cs(n_{2,2}, nw : n : ne_{1,2})$ =1-0.02=0.98.*
- *maps M_7 and M_9: $cs(n_{2,2}, nw : n_{1,2})$ =1-0.01=0.99.*
- *maps M_8 and M_9: $cs(nw : n : ne_{1,2}, nw : n_{1,2})$ =1-0.015=0.985.*

In this case, we get the expected result. $\square$

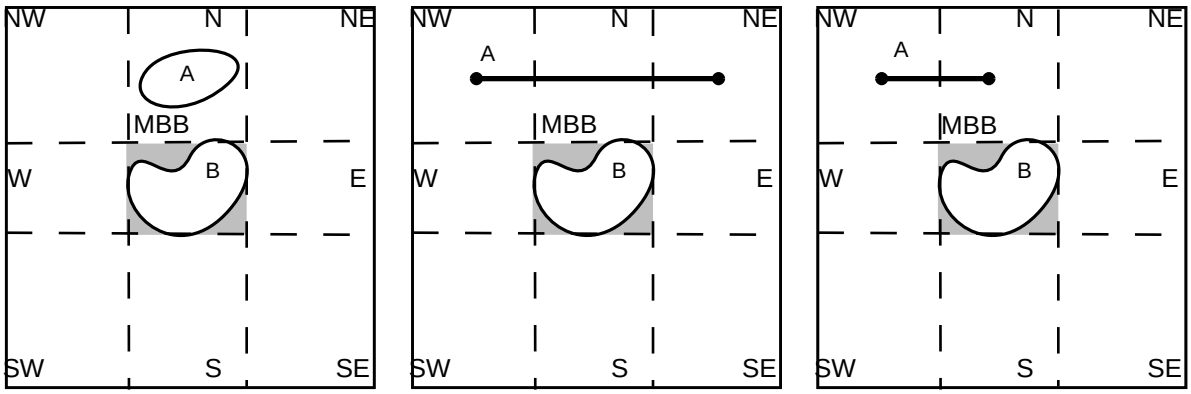


Figure 3.4: Consistency between direction relations.

3.4 Consistency and Similarity of Geo-Spatial Relations

In the context of multi-resolution maps, the problem arises of establishing whether such maps are consistent, i.e., they represent common objects in a consistent way. Informally, by considering qualitative information, two maps are consistent when, given any pair of common features, they share the same or similar geo-spatial relationships in both maps. In particular, we focus on topological and direction consistency, i.e., consistency with respect to topological and direction relationships existing between map objects. Different types of consistency can be defined: equality-based (eq-based) consistency and similarity-based (sim-based) consistency. *Eq-based consistency* requires that the same geo-spatial relationships exist between each pair of common features in two maps.

Definition 12 (*Eq-based consistency*) Let $\chi_{d_1,d_2} \in \mathcal{T}_d \cup \mathcal{C}_d$. f_1 and f_2 in map M_1 are *eq-based consistent* with f_1 and f_2 in map M_2, if $f_1 \chi_{d_1,d_2} f_2$ holds in both M_1 and M_2. M_1 and M_2 are eq-based consistent if, for any pair of features $(f_1, f_2) \in (M_1 \cap M_2)^2$, f_1 and f_2 in map M_1 are eq-based consistent with f_1 and f_2 in map M_2. □

Since topological relations are not defined for all possible pairs of dimensions (see Table 3.1), it is simple to note that eq-based consistency cannot always be guaranteed. For example, `overlap` is defined between pairs of regions and pairs of lines, but it is not defined between a line and a region. A similar situation arises for multi-tile relations, which are not defined when the target object is a point.

From this consideration it follows that geo-spatial relationship equality is a too strong criterion for defining consistency. Such criterion can however be relaxed by considering similarity instead of equality between relations. The new notion of consistency, that we call *sim-based consistency*, is always defined and requires that geo-spatial relationships between two features in two different maps are not necessarily equal but the most similar ones. Sim-based consistency seems the most reasonable choice in real situations dealing with multi-resolution, where eq-based consistency cannot always be guaranteed.

In order to formally define sim-based consistency, we rely on the similarity functions we have defined in Sections 3.2.2 and 3.3.2. More precisely, sim-based consistency can be defined by requiring that geo-spatial relationships between pairs of features in two distinct maps must be the most similar ones, according to the introduced similarity functions.

Definition 13 (*Sim-based consistency*) Let $\chi^1_{d_1,d_2}, \chi^2_{d_3,d_4} \in \mathcal{SR}$, where $\mathcal{SR}$ is either $\mathcal{T}_d$ or $\mathcal{C}_d$. Let f_1 and f_2 be two features appearing in map M_1 with dimensions (d_1, d_2) and in map M_2 with dimensions (d_3, d_4). f_1 and f_2 in map M_1 are *sim-based consistent* with f_1 and f_2 in map M_2 if $f_1\chi^1_{d_1,d_2}f_2$ holds in M_1, $f_1\chi^2_{d_3,d_4}f_2$ holds in M_2 and $\overline{s}(\chi^1_{d_1,d_2}, \chi^2_{d_3,d_4})$ coincides either with $max\{\overline{s}(\chi^1_{d_1,d_2}, \chi^3_{d_3,d_4})|\chi^3_{d_3,d_4} \in \mathcal{SR}_{d_3,d_4}\}$ *or* $max\{\overline{s}(\chi^2_{d_3,d_4}, \chi^3_{d_1,d_2})|\chi^3_{d_1,d_2} \in \mathcal{SR}_{d_1,d_2}\}$. where $\overline{s}$ is either ts or cs, depending on $\mathcal{SR}$. M_1 and M_2 are sim-based consistent if for any pair of features $(f_1, f_2) \in (M_1 \cap M_2)^2$, f_1 and f_2 in map M_1 are sim-based consistent with f_1 and f_2 in map M_2. □

We denote by $C^{\mathcal{T}}_{d_3,d_4}(\theta^1_{d_1,d_2})$ the set of all relations in $\mathcal{T}_{d_3,d_4}$, that are sim-based consistent with $\theta^1_{d_1,d_2}$ and $C^{\mathcal{C}}_{d_3,d_4}(\gamma^1_{d_1,d_2})$ the set of all relations in $\mathcal{C}_{d_3,d_4}$ which are sim-based consistent with $\gamma^1_{d_1,d_2}$. Sim-based consistency satisfy several useful properties.

Proposition 3 *Sim-based consistency satisfies the following properties:*

1. *it is a many-to-many relationship, i.e., cardinality of* $C^X_{d_3,d_4}(\chi^1_{d_1,d_2})$, $X \in \{\mathcal{T}, \mathcal{C}\}$, *may be greater than one;*
2. *it is symmetric, i.e.,* $\chi^2_{d_3,d_4} \in C^X_{d_3,d_4}(\chi^1_{d_1,d_2})$ *if and only if* $\chi^1_{d_1,d_2} \in C^X_{d_1,d_2}(\chi^2_{d_3,d_4})$, $X \in \{\mathcal{T}, \mathcal{C}\}$;
3. *it is reflexive, i.e.,* $C^X_{d_1,d_2}(\chi^1_{d_1,d_2}) = \{\chi^1_{d_1,d_2}\}$ *and* $cs(\chi^1_{d_1,d_2}, \chi^1_{d_1,d_2}) = 1$ $(ts(\chi^1_{d_1,d_2}, \chi^1_{d_1,d_2}) = 1)$. □

Concerning item (1), it can be shown that for example $C^{\mathcal{T}}_{1,1}(o_{2,2})$=$\{o_{1,1}, r_{1,1}\}$ see Table 3.3.

Depending on the type of the considered geo-spatial relations, there exists different relationships between eq-based and dist-based consistency notions, as pointed out by the following proposition.

Proposition 4 *The following properties hold:*

1. *For all topological similarity functions satisfying properties **P1** and **P2**, eq-based consistency implies sim-based consistency, i.e.,* $\theta^1_{d_3,d_4} \in C^{\mathcal{T}}_{d_3,d_4}(\theta^1_{d_1,d_2})$*;*
2. *For the topological similarity function introduced in Section 3.2.2, eq-based consistency does not implies sim-based consistency when* $\theta^1_{d_1,d_2} = touch$ *and* $(d_1, d_2) = (*, 0)$ *and* $(d_3, d_4) = (0, *)$ *with* $*$ *means any value in* 1, 2*.*
3. *For the topological similarity function introduced in Section 3.2.3, eq-based consistency does not implies sim-based consistency when* $\theta^1_{d_1,d_2} = touch$ *and* $(d_1, d_2) = (*, 0)$ *and* $(d_3, d_4) = (0, *)$ *with* $*$ *means any value in* 1, 2*.*
4. *Considering direction relationships: (a)* $C^{\mathcal{C}}_{d_3,d_4}(\gamma^1_{d_1,d_2}) = \{\gamma^1_{d_1,d_2}\}$ *when* $d_3 \neq 0$ *or* $\gamma^1_{d_1,d_2}$ *is a single-tile relation; (b)* $C^{\mathcal{C}}_{d_3,d_4}(\gamma^1_{d_1,d_2}) \subseteq s(\gamma^1_{d_1,d_2})$ *when* $d_3 = 0$ *and* $\gamma^1_{d_1,d_2}$ *is a multi-tile relation.*

Proof sketch.

Item (1) directly follows from the properties **P1** and **P2**. For both item (2) and (3) directly follows from Tables 3.2, 3.3, and Tables 3.4, 3.5 respectively. This strange behavior is probably due to boundary information, quite relevant for the *touch* relationship, that are lost when transforming a region into a point. Item (4a) follows from the fact that, under the stated condition, eq-based consistency can always be defined and therefore $cs(\gamma^1_{d_1,d_2}, \gamma^1_{d_3,d_4}) = 1$. Moreover, it is easy to show that the similarity value 1 can only be obtained when the direction relations coincide. Item (4b) follows from the fact that under the stated condition, $\gamma^1_{d_1,d_2} \notin \mathcal{C}_{d_3,d_4}$ and relations in $s(\gamma^1_{d_1,d_2})$ obviously minimize the distance. □

3.5 Summary

In this chapter, we have presented a complete framework to assess consistency and similarity between multi-resolution geo-spatial datasets using qualitative information, that is, geo-spatial relations and, in particular, topological and direction relations.

First of all, we have extended existing models for representing topological and direction relations and the existing similarity functions for both direction and topological relations in order to cope with multi-resolution maps.

After that, we have presented a unified framework for assessing topological and direction consistency between multi-resolution maps, exploiting the similarity functions we have

defined. Two different types of consistency have been presented, equality-based consistency and similarity-based consistency. Moreover, only for topological relations we have defined another similarity function for the relaxed query processing. This function addresses two important properties: (i) it assigns different similarity values to different pairs of topological relations, and (ii) it does not assign `disjoint` as the most similar relation to predicates different from `disjoint` itself (excluding the topological relation *touch*).

In Chapter 4, we will present the relaxed geo-spatial selection and join operators based on the concepts of consistency introduced here providing also three different semantics. We, finally, present and discuss also the query processing algorithms for both operators.

Chapter 4

Relaxed Geo-Spatial Operators

In the last years, there has been a rapid evolution of environments and applications that has radically modified query processing over data collections. Indeed, in data integration applications, Web services, data streams, P2P systems, hosting, just to cite a few, data characteristics are quite variable and unpredictable. As a consequence, the user may not always be able to specify the query in a complete and exact way since she may not know all the characteristics of data to be queried, even if data come from just one single source (possibly, because such characteristics may change during query execution, as in mashup applications). Thus, the quality of the obtained result, in terms of completeness and accuracy, may be reduced, since interesting objects may not be returned. This problem can be partially avoided by relaxing search conditions or approximating their evaluation. Several approaches have been proposed so far for relaxing or approximating queries in the relational and XML contexts. Nowadays, due to the rapid diffusion on the net of geo-spatial data collections, relaxation and approximation problems are becoming quite relevant also for geo-spatial applications.

As pointed out in Chapter 2 query relaxation constitutes one of the hot topics in query processing, but, as far as we know, few approaches have been proposed so far for defining and efficiently executing relaxed geo-spatial queries considering in particular geo-spatial relations. Moreover, although the importance of topological relations is well known in different research areas (e.g., spatial reasoning, cognitive sciences, etc...), most of the existing work on geo-spatial processing has concentrated on topological relations for pairs of regions, even if various sets of topological relations have been defined for objects of any type. As presented in Chapter 2 only few proposals exist for efficiently executing queries based on generic topological relations, and even less address the problem of relaxation of geo-spatial predicates.

In this chapter we address the relaxation problem for spatial queries and we provide so-

lutions for relaxing selection and join conditions based on topological relations, whatever is the type of objects they are applied on. The proposed relaxed operators improve the quality, in terms of completeness and accuracy, of the obtained result with respect to non-relaxed ones. For both operators, we present also different semantics addressing different relaxation issues.

As a second contribution, we present query processing algorithms for the execution of the defined relaxed topological selection and join operators. The proposed algorithms rely on the usage of the R-tree index structure and are based on a branch and bound approach, which discards the visit of some R-tree sub-trees that cannot produce further results. The pruning condition is defined by using a topological similarity function and some compatibility rules, first proposed in [PTSE95] for regions, here extended to pairs of spatial objects of arbitrary dimension.

The content of the chapter is organized as follows. We first present, in Section 4.1, the basis of our relaxation strategy; then, in Section 4.2, the motivations we have taken into account in the definition of relaxed operators are discussed. Then, we start presenting in Section 4.3 the general and formal definition of both relaxed selection and join operators based on topological conditions. We present three different semantics both for selection and join operators in Section 4.4. Finally, in Section 4.5, we illustrate the proposed query processing algorithms for the execution of relaxed operators under the proposed semantics.

4.1 Relaxation Strategy

Since not all topological predicates are defined for each pair of object types (for example, *overlap* is not defined between a region and a line), two distinct types of selection and join conditions are considered for relaxation:

- *Well-defined conditions*: they rely on topological relations which are defined for the objects used in the condition. Asking for all rivers, represented as polygons, which *overlap* a given region is a well-defined condition.
- *Definable conditions*: definable conditions are non well-defined because they rely on a topological relation that is not defined for the spatial dimension of the considered feature types, but they would become well-defined by changing such dimensions. Asking for all rivers, represented as lines, which *overlap* a given region is a definable condition since *overlap* is not defined between a line and a region, but it is defined between a pair of regions. Definable conditions often arise in environments where the user does not know, for some application reason, the feature type dimensions (the dimension of rivers, in the example above); definable conditions thus usually depend on a lack of knowledge. In those cases, it seems reasonable to assign a semantics to

the query anyway, by rewriting the query condition using the most similar topological relations defined for the dimension of features in the available data. For this reason, we say that the condition is *definable* since the syntax error can be "recovered" by query rewriting.

Notice that join conditions can be either well-defined or definable. On the other hand, selection conditions may be *incorrect* when the query relation is not compatible with the query object dimension. For example, when O is a point, condition ft *overlap/cross* O is incorrect since *overlap* and *cross* are not defined for query objects that are points, whatever dimension we have at disposal for feature type ft. Incorrect conditions are not taken into account in the considered query relaxation process since the error in this case does not depend on a missing information but just on a wrong query specification (since the user must be aware of the dimension of O, which is chosen by her).

The relaxation strategy we propose is based on: (i) a relaxation predicate $rp()$, that applies a certain degree of relaxation in selecting result objects (or objects pairs); (ii) a similarity function among topological relations, which provides a relaxation measurement. Different definitions for the relaxation predicate provide different semantics for the relaxed operators. Different definitions for the similarity function generate different result sets during query execution. In this chapter we propose three semantics for $rp()$. The Best Fit semantics applies the minimum amount of relaxation to the query condition in order to return a non empty answer. Thus, it models a variation of a top-1 query dealing with spatial relations. The Threshold semantics relaxes the topological query up to a certain fixed limit, depending on system parameters. While under the Threshold semantics relaxation is always applied and, if required, can be hardcoded, in the Best Fit semantics relaxation depends on the input dataset and is applied only if no spatial objects satisfy the query condition (thus, guaranteeing a non empty relaxed result). The Nearest Neighbor semantics, defined only for definable conditions, relaxes the topological condition with its closest topological relations.

4.2 Motivating Example

In order to understand the need for the approach we develop, let us consider a database containing the provinces, the rivers, and the railways modeled by three feature types *Province* (*Pr*), *River* (*Rv*), and *Railway* (*Rl*), of the region of Northern Italy around Genoa (called Liguria), as shown in Figure 4.1. For some application reasons, suppose that different surveys have been carried out and data are organized into four distinct maps M_{Pr}, M_{Rl}, M_{MRv}, and M_{Rv}, containing respectively: provinces, represented as regions; main railways, represented as lines; main rivers, represented as regions; minor rivers, represented as lines. Railways and main rivers are labeled in the figure as $R*$ and $F*$, respectively; minor rivers are not labeled. Provinces are labeled by using their code.

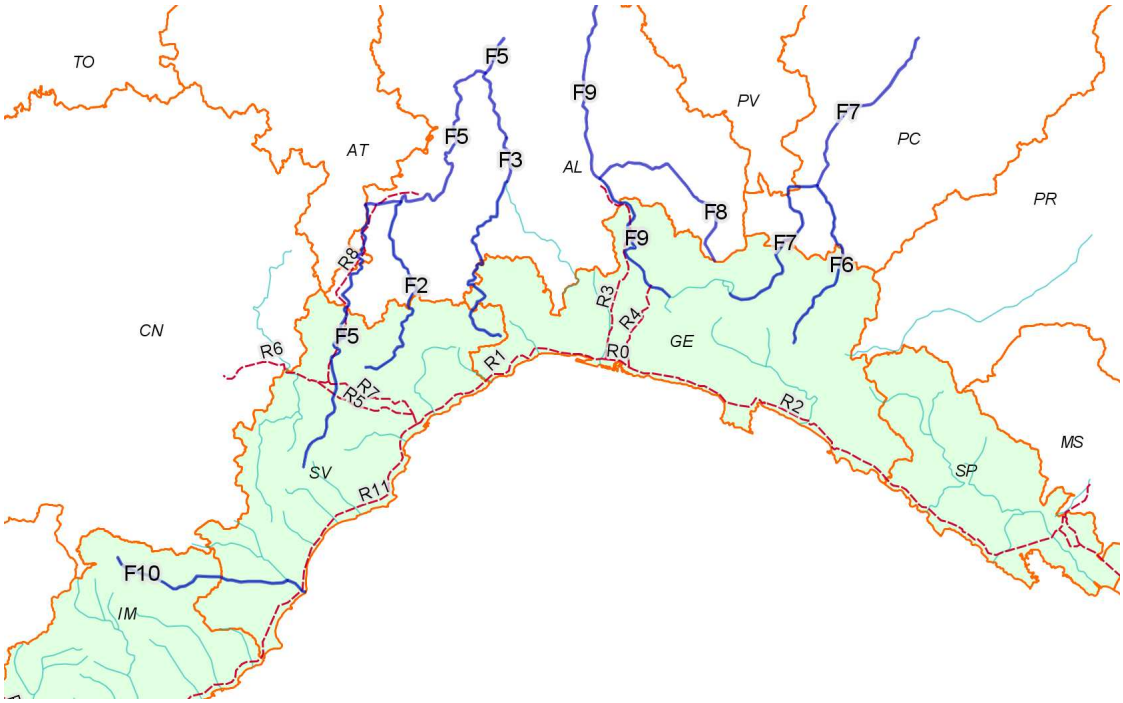


Figure 4.1: Datasets used in the running example: Provinces of Liguria (IM,SV,GE,SP) are filled in light gray; main rivers are filled in dark gray and labeled $F*$; minor rivers are lines in light gray; railways are dashed lines and are labeled $R*$.

Query **Expression**	**Query result**		
	Non relaxed semantics	**Best Fit relaxed semantics**	**Threshold relaxed semantics**
Q_1 $\sigma_{<Rv \text{ overlap } \mathbf{L}>}(M_{MRv})$	main rivers which overlap Liguria, without river $F10$ (it is inside Liguria, but it also touches its boundary)	same result as non relaxed semantics, since there exist rivers that overlap Liguria	with a certain amount of relaxation, also $F10$ will be returned since it is inside Liguria and touches the boundary of Liguria (i.e., it is *covered by* Liguria)
Q_2 $\sigma_{<Rv \text{ cross } \mathbf{L}>}(M_{Rv})$	minor rivers that `cross` the boundary of Liguria (notice that they are only those that go North)	same result as non relaxed semantics, since there are rivers that cross Liguria	with a certain amount of relaxation, also rivers that are covered by or are inside Liguria will be returned
Q_3 $\sigma_{<Rv \text{ in } \mathbf{L}>}(M_{MRv})$	the empty set (river $F10$ is covered by Liguria, since it touches the boundary of Liguria)	main rivers that are covered by Liguria (thus, river $F10$ will be returned)	with a certain amount of relaxation, also rivers that are covered by or overlap Liguria
Q_4 $\sigma_{<Rl \text{ overlap } \mathbf{L}>}(M_{Rl})$	syntactic error	it can be rewritten into $\sigma_{<Rl \text{ cross } \mathbf{L}>}(M_{Rl})$	
Q_5 $M_{Pr} \bowtie_{<Pr \text{ cross } Rv>} M_{Rv}$	pairs (p, r) representing a province p with a minor river r it crosses	same result as non relaxed semantics, since there are provinces that cross rivers	with a certain amount of relaxation, the relaxed query includes the pairs (p, r) representing a province p with a minor river r it covers or contains
Q_6 $\sigma_{Reg='Liguria'}(M_{Pr})$ $\bowtie_{<Pr \text{ contain } Rv>} M_{MRv}$	the empty set, since no pair (p, r), representing the Liguria province p with a major river r it contains, exists	pairs (p, r) representing the Liguria province p with a major river r it covers	with a certain amount of relaxation, also the pairs (p, r) representing the Liguria province p with a major river r it covers or overlaps
Q_7 $M_{MRv} \bowtie_{<Rv \text{ in } Rl>} M_{Rl}$	syntactic error	it can be rewritten into $M_{MRv} \bowtie_{<Rv \text{ cover } Rl>} M_{Rl}$	

Table 4.1: Query examples. $\mathbf{L} = Liguria$, $Rv = Rivers$, $Rl = Railways$ and $Pr = Provinces$.

Now suppose the user is interested in the rivers that pass through the provinces of Liguria, and suppose that she has obtained with previous computation the polygon representing Liguria (*Liguria*). She may specify one of the queries represented in Table 4.2. Some of such queries return the expected result (Q_1, Q_2 and Q_5), since there exist objects (or object pairs) that satisfy the specified topological relation. Others, as Q_3 and Q_6, are syntactically correct but no objects (or object pairs) in the database satisfy the given condition, even if objects (or object pairs) that satisfy a very similar condition exist. Finally, queries Q_4 and Q_7 generate a syntactic error since the used topological relation is not defined for the feature type dimensions at hand.

We claim that all the considered queries may get benefits from a relaxed execution. To this purpose, we introduce three different semantics: (i) the Best Fit semantics (BF), which applies the minimum amount of relaxation to the query condition in order to return a non empty answer; (ii) the Threshold semantics (Thr), which relaxes the topological query up to a certain fixed limit, depending on system parameters. and (iii) the Nearest Neighbor semantics (NN), which relaxes the topological condition with its closest topological relations, used also for definable conditions.

The usage of such semantics in executing queries $Q_1 - Q_7$ leads to the following relaxed results:

- The usage of the Best Fit semantics in queries Q_1, Q_2 and Q_5 does not extend the non relaxed result, since it is not empty (and therefore the relaxed result coincides with the non relaxed one). On the other hand, under the Threshold semantics, the result is extended anyway by relaxing the requested topological relation within a given threshold. For example, in Q_1 and Q_2, which require rivers that `overlap` and `cross` Liguria, respectively, also rivers that are `coveredby` *Liguria* (i.e, that are inside and touch the boundary of *Liguria*), for Q_1, or are `coveredBy` or `in` *Liguria*, for Q_2, could be returned if the similarity between `overlap` and `coveredby`, for Q_1, and between `cross` and `coveredby/in`, for Q_2, is greater than the specified threshold. Similarly, the non relaxed result for query Q_5 does not include many rivers that go south, since the Liguria province covers them, while a `cross` is requested by the join operator. An approximate execution of Q_5 based on the Threshold semantics could eventually add also these rivers, since `cover` relation is quite similar to the `cross` and `contain` relations.

- Since queries Q_3 and Q_6 produce an empty result, their execution under the Best Fit semantics ends up with a result extension, relaxing the query relations with their more similar ones, according to some similarity function. In particular, relation `in` in Q_3 could be relaxed with `coveredby`, while relation `contain` in Q_6 with `cover`. Further relations could be considered by applying the Threshold semantics.

- Queries Q_4 and Q_7 generate a syntactic error, but they can be 'recovered' through

rewriting, according to some similarity function and to a particular semantics among Best Fit, Threshold and also Nearest Neighbor, the wrong topological query predicate into a topological relation defined for the objects in the considered maps.

In Q_4, the error occurs since `overlap` is not defined between a line - railways - and a region - the *Liguria* object. Q_4 can be 'recovered', considering the different semantics as follows: (i) under the Best Fit semantics `overlap` is substitued with its more similar topological relations, according to some similarity function, actually present among objects in the considered maps, such as, `cross` in this case; (ii) under Threshold semantics `overlap` can be substituted by `cross` and other further relations depending on the used threshold; (iii) while the Nearest Neighbor semantics relaxes the query relations with their more similar ones defined for the considered objects, in particular, `overlap` is substituted with `cross` defined between a line and a region.

In a similar way, Q_7 can be 'recovered' by replacing the `in` relation with the `cover` relation. The selection (join) condition of Q_4 (Q_7) is an example of *definable* condition, since it would become a well-defined condition in case the railways were regions.

With respect to the example above, the relaxed operators we present in the chapter thus guarantee, under the Best Fit semantics, that queries Q_3 and Q_6 produce a non empty result. Under the Threshold semantics, the proposed relaxed operators allow to extend the result of queries Q_1, Q_2, Q_3, Q_5 and Q_6. Also for definable topological conditions we define the Best Fit, Threshold and the Nearest Neighbor semantics, providing a computation strategy for queries Q_4 and Q_7. In all cases, the result extension will return to the user potentially interesting results which would not have been determined by using non relaxed operators, thus potentially improving query satisfaction.

4.3 Relaxed Geo-Spatial Operators

In the following, we present relaxed selection and join operators based on topological conditions. As already pointed out, conditions can be either:

- *well-defined*, if they rely on topological relations which are defined for the objects used in the condition;
- *definable*, if they are non well-defined because they rely on a topological relation that is not defined for the spatial dimension of the considered feature types, but they would become well-defined by changing such dimension;
- *incorrect* conditions, if they are neither well-defined nor definable.

We denote with $\mathcal{SC}_{MS}$ and $\mathcal{JC}_{MS_a,MS_b}$ the set of all well-defined or definable selection conditions and join conditions, respectively, over map schemas MS, MS_a, and MS_b.

While we retain the syntax of traditional selection and join operators, the assigned semantics is different, in order to retrieve not only features that precisely satisfy the given condition but also those features which approximate, to some extent, the desired result.

As already introduced, the semantics of relaxed topological operators is provided in a parametric way, with respect to: (i) a *topological similarity function* ts, as defined in Section 3.2.3, which is used for computing ranking values for features; each ranking value quantifies the similarity of the feature (the pair of features) with respect to the specified query condition, based on the similarity of the topological predicate satisfied by the feature with respect to the query object (or satisfied by the pair of features for join) and the query predicate. Of course, if $\theta^q = \theta^m$, then $ts(\theta^m, \theta^q) = 1$; (ii) a *relaxation predicate* rp, which establishes whether a feature (a pair of features) has to be returned as result, depending on the computed ranking value. Thus, it specifies which similarity values are acceptable, with respect to the input maps and the given condition.

We decide to restrict the query conditions to single topological conditions, that is to condition involving only one topological relation (we will formal define them for selection and join in Section 4.3.1 and Section 4.3.2 respectively); however, in Chapter 6 we briefly present and discuss a possible approach for complex topological conditions (i.e., conditions involving more than one topological relation).

In the reminder of this section, we present both relaxed selection and join operators (Section 4.3.1 and Section 4.3.2 respectively).

4.3.1 Relaxed Spatial Selection Operator

In the following, we present a relaxed geo-spatial selection operation based on single topological conditions, for both well-defined and definable selection conditions.

Definition 14 (*Single topological selection condition*) Let $O \in OBJ$, let ft be a feature type defined in a map schema MS. Let $\theta^q \in \mathcal{T}$.

Then, $C \equiv< ft\theta^q O >$ is a *selection condition*. If $\theta^q \in \mathcal{T}_{d(ft),d(O)}$, C is *well-defined*; if $\theta^q \notin \mathcal{T}_{d(ft),d(O)}$ but there exists $d \in DIM$ such that $\theta^q \in \mathcal{T}_{d,d(O)}$, C is *definable*.

All the other conditions are *incorrect* and no semantics is provided for them. θ^q is the *topological query relation*, O is the *query object*, and ft *query feature type*. □

As pointed out before, the definition of the relaxed topological selection operator relies on the usage of a relaxation predicate, formally defined as follows.

Definition 15 (*Selection relaxation predicate for well-defined conditions*) The selection relaxation predicate for well-defined condition is defined as:

$rp : \mathcal{M}_{\mathcal{MS}a} \times \mathcal{SC}_{MS_a} \times [0,1] \rightarrow Bool$. □

Relaxed topological operators compute the ranking value according to ts for each feature. Then, only features that, when considering the computed ranking value and the actual map, make rp true are returned as result. The semantics of relaxed selection operators can therefore be defined as follows.

Definition 16 (*Semantics of relaxed selection operator based on well-defined conditions*) Let ts be a topological similarity function, rp a relaxation predicate. The semantics of the relaxed selection operator can be defined as follows:

$\sigma_{<ft\theta^q O>}(M) = \{f | f \in ext(ft, M) \wedge f\theta^m O \wedge rp(M, ft\theta^q O, ts(\theta^m, \theta^q))\}$. □

Based on the previous definition, the relaxed selection operator returns all the features f, instances of the feature type ft, such that the topological similarity between the topological relation θ^m satisfied by f and O and the topological relation θ^q requested by the query satisfies in M the relaxation predicate $rp()$.

Example 11 *Consider the topological similarity function ts presented in Table 3.4 and Table 3.5 and the motivating example presented in Section 4.2. Now suppose that* $rp(M_{MRv}, Rv$ `overlap` $Liguria, \alpha)$ *is true if and only if* $\alpha > 0.7$. *Since* $ts($`overlap`, `coveredby`$) = 0.715$, $Q_1 \equiv \sigma_{<Rv\ \texttt{overlap}\ Liguria>}(M_{MRv})$, *under the relaxed semantics returns not only rivers which* `overlap` *Liguria (namely, rivers* $F5, F2, F3, F9, F7, F6$*), but also rivers which* `cover` *(of course, none) or are* `coveredby` *Liguria. Thus, also river* $F10$ *is returned as result (see Figure 4.1).*

In order to define a semantics for relaxed topological operators based on definable selection conditions, the basic idea is that of replacing the definable condition with a disjunction of well-defined conditions, which are most similar to the given one. Such conditions can be determined as follows:

- We first determine the feature type dimensions d_a closest to $d(ft_a)$ for which the query relation is defined. Notice that, since the condition is definable, we are sure that such dimensions exist. In case more than one choice exists, we take the one corresponding to the highest dimension.

- We then determine the set of topological relations in $\mathcal{T}_{d(ft_a),d(O)}$ closest to $\theta^q_{d_a,d(O)}$, with respect to the considered similarity function. We call this set *replacing set*. That is, we determine the closest topological relation to that specified in the query for the feature type dimensions at hand.

Formally:

Definition 17 (*Replacing set for a definable selection condition C ($\mathcal{F}_s(C)$)*) Let $C \equiv< ft_a\ \theta^q\ O >\in \mathcal{SC}_{MS_a}$. Let d_a defined as above. The replacing set $\mathcal{F}_s(C)$ is defined as follows:

$$\mathcal{F}_s(C)= \{\theta | \theta \in \mathcal{T}_{d(ft_a),d(O)} \wedge ts(\theta^q_{d_a,d(O)}, \theta) = max\{ts(\theta^q_{d_a,d(O)}, \psi) | \psi \in \mathcal{T}_{d(ft_a),d(O)}\}\}$$

Given a topological relation θ^m, a replacing set $\mathcal{F}_s(C)$, and a topological similarity function ts, we define $ts(\theta^m, \mathcal{F}_s(C))$ as $max\{\bigcup_{\theta_i \in \mathcal{F}_s(C)}\{ts(\theta^m, \theta_i)\}\}$. □

The replacing set thus represents the set of topological relations that can be used to approximate the one specified in the definable condition. Similarly to what have been done for well-defined selection conditions, semantics for definable conditions relies on the usage of a relaxation predicate rp. All the features that satisfy such predicate are returned as result. In this case, however, as ranking value to be considered by the relaxation predicate in the definition of relaxed operators, we take $ts(\theta, \mathcal{F}_s(C))$, i.e., the maximum similarity value among those computed with respect to each relation in the replacing set.

Relaxed selection operator for definable conditions can now be defined as follows:

Definition 18 (*Semantics of relaxed selection operator based on definable conditions*) Let ts be a topological similarity function, rp a relaxation predicate. Let $C \equiv< ft_a\ \theta^q\ O >\in \mathcal{SC}_{MS_a}$. Relaxed selection operator can be defined as follows:

$$\sigma_C(M) = \{f | f \in ext(ft, M) \wedge f\theta^h O \wedge rp(M, C, ts(\theta^h, \mathcal{F}_s(C)))\}.$$

Example 12 *Consider the query Q_4, in Table 4.2. Since the dimension of features in M_{Riv} is 1 and the dimension of query object is 2, then, the query contains a syntactic error, since* `overlap` *it is not defined for the dimensions (1,2).*

We first determine the feature type dimension d_{def} which is closest to 1*, for which the query relation* $\texttt{overlap}_{\texttt{d}_{\texttt{def}},\texttt{2}}$ *is defined. Such dimension is 2. We then determine the set $\mathcal{F}_s(C)$ of topological relations $\theta_{1,2}$ closest to* $\texttt{overlap}_{2,2}$*, with respect to the considered similarity function, based on the Tables 3.4, 3.5, $\mathcal{F}_s(C)$=*$\{\texttt{cross}_{1,2}\}$.

□

4.3.2 Relaxed Geo-Spatial Join Operator

The geo-spatial join operator can be defined similarly to the selection operator. In the following, we present the formalization for join operator based on single well-defined and definable topological conditions.

Definition 19 (*Single topological join condition*) Let ft_a, ft_b be feature types defined in map schemas MS_a, MS_b, respectively. Let $\theta^q \in \mathcal{T}$.

If $\theta^q \in \mathcal{T}_{d(ft_a),d(ft_b)}$, c is well-defined; if the previous condition is not satisfied, the condition is definable.[1]

θ^q is the *topological query relation*, and ft_a, ft_b *query feature types.* □

The signature of the join relaxation predicate for well-defined conditions, used to formalized the join operator, is defined as follows.

Definition 20 (*Join relaxation predicate for well-defined condition.*) The join relaxation predicate for well-defined conditions is defined as:

$rp : \mathcal{M}_{MS_a} \times \mathcal{M}_{MS_b} \times \mathcal{JC}_{MS_a,MS_b} \times [0,1] \rightarrow Bool.$ □

The relaxed join operator for well-defined condition is defined as follows.

Definition 21 (*Semantics of relaxed join operator for well-defined condition*) Let ts be a topological similarity function, rp a relaxation predicate. The relaxed join operator can be defined as follows:

$M_a \bowtie_{<ft_a\theta^q ft_b>} M_b = \{(f_a, f_b) | f_a \in ext(ft_a, M_a) \wedge f_b \in ext(ft_b, M_b) \wedge f_a\theta^m f_b \wedge rp(M_a, M_b, ft_a\theta^q ft_b, ts(\theta^m, \theta^q))\}.$ □

The relaxed join operator returns all the feature pairs (f_a, f_b), instances of ft_a in M_a and of ft_b in M_b respectively, such that the topological similarity between the topological relation θ^m satisfied by f_a and f_b and the topological relation θ^q requested by the query satisfies in M_a, M_b the relaxation predicate rp.

Example 13 *Consider the topological similarity function for relaxation context defined in Section 3.2.3 and the maps M_{MRiv}, M_{Pro} presented in Example 4.2. Assume that*

[1] Notice that for join conditions, since they do no depend on a fixed input object, it is always possible to find $d_a, d_b \in DIM$ such that $\theta^q \in \mathcal{T}_{d_a,d_b}$.

$rp(M_{MRiv}, M_{Pro}, Rv \quad \theta^q \quad Pro, \alpha)$ holds if α >0.7. Now, consider the query $\bowtie_{<Rv\ \texttt{overlap}\ Pro>}(M_{MRiv}, M_{Pro})$. Since we admit a relaxation of 0.7 of the `overlap` *relation, we are searching for* `overlap`*, but also for* `coveredby` *and* `cover` *(see Tables 3.4, 3.5.) Indeed,* ts(`overlap`, `overlap`)*=1 and* ts(`overlap`, `coveredby`) $=ts$(`overlap`, `cover`)*=0.715; thus, both the similarity values satisfy the relaxation predicate. The results set contains the feature pairs that satisfy the following relations:* $o_{2,2}(F10, IM)$,$o_{2,2}(F10, SV)$, $o_{2,2}(F5, SV)$,$o_{2,2}(F5, AL)$, $o_{2,2}(F2, SV)$, $o_{2,2}(F3, AL)$, $o_{2,2}(F3, GE)$, $o_{2,2}(F3, SV)$, $o_{2,2}(F9, GE)$, $o_{2,2}(F9, AL)$, $o_{2,2}(F11, GE)$, $o_{2,2}(F6, GE)$, $o_{2,2}(F11, PC)$, $o_{2,2}(F6, PC)$, $o_{2,2}(F9, AL)$, $b_{2,2}(F1, AL)$, $b_{2,2}(F12, AL)$,$b_{2,2}(F8, AL)$, $b_{2,2}(F7, PC)$,$b_{2,2}(F1, AL)$, $b_{2,2}(F13, GE)$, *where* F_i *are rivers and* { *IM, SV, GE, AL, OC* } *are provinces.* □

We present now the join relaxation predicate for definable join conditions and, based on it, the relaxed join operator for definable join conditions.

Also in this case as for selection, the basic idea is to substitute the definable join condition $< ft_a \theta^q ft_b >$ with a disjunction of well-defined conditions, which are most similar to the given one. Such conditions are determined as follows:

- We first determine the feature type dimensions d_a, d_b closest to $d(ft_a), d(ft_b)$ for which the query relation is defined. Notice that, since the condition is definable, we are sure that such dimensions exist. In case more than one choice exists, we take the one corresponding to the highest dimension.
- We then determine the set of topological relations in $\mathcal{T}_{d(ft_a),d(ft_b)}$ closest to $\theta^q_{d_a,d_b}$, with respect to the considered similarity function. We call this set *replacing set*. That is, we determine the closest topological relation to that specified in the query for the feature type dimensions at hand.

Formally, the replacing set for join operator is defined as follows:

Definition 22 (*Replacing set for a definable join condition ($\mathcal{F}_j(C)$)*) Let $C \equiv < ft_a\ \theta^q\ ft_b > \in \mathcal{JC}_{MS_a,MS_b}$. Let d_a and d_b defined as above. The replacing set $\mathcal{F}_j(C)$ is defined as follows:

$$\mathcal{F}_j(C) = \{\theta | \theta \in \mathcal{T}_{d(ft_a),d(ft_b)} \wedge ts(\theta^q_{d_a,d_b}, \theta) = max\{ts(\theta^q_{d_a,d_b}, \psi) | \psi \in \mathcal{T}_{d(ft_a),d(ft_b)}\}\}.$$

Given a topological relation θ^m, a replacing set $F_j(C)$, and a topological similarity function ts, we define $ts(\theta^m, F_j(C))$ as $max\{\bigcup_{\theta_i \in \mathcal{F}_j(C)}\{ts(\theta^m, \theta_i)\}\}$. □

Similarly to what have been done for well-defined join conditions, semantics for definable relaxed operators relies on the usage of a relaxation predicate rp. All the features that satisfy such predicate are returned as result. In this case, however, as ranking value to be considered by the relaxation predicate in the definition of relaxed operators, we take $ts(\theta, \mathcal{F}_j(C))$, i.e., the maximum similarity value among those computed with respect to each relation in the replacing set.

Relaxed join operator for definable conditions can now be defined as follows:

Definition 23 (*Relaxed join operator for definable condition*) Let ts be a topological similarity function, rp a relaxation predicate. Let $C \equiv< ft_a\ \theta^q\ ft_b >\in \mathcal{JC}_{MS_a,MS_b}$. The relaxed join operator can be defined as follows:

$$M_a \bowtie_C M_b = \{(f_a, f_b) | f_a \in ext(ft_a, M_a) \wedge f_b \in ext(ft_b, M_b) \wedge f_a \theta^h f_b \wedge rp(M_a, M_b, C, ts(\theta^h, \mathcal{F}_j(C)))\}. \quad \square$$

Example 14 *Now consider query Q_7. Also in this case, condition C is non well-defined, since in is not defined between a region, representing a main river, and a line, used for describing a railway.* $\texttt{in}_{2,2}$ *is however defined. The replacing set $\mathcal{F}_j(C)$ thus contains all relations in $\mathcal{T}_{2,1}$ closest to* $\texttt{in}_{2,2}$. *Based on the similarity function presented in Section 3.2.3, we have that $\mathcal{F}_j(C) = \{\texttt{cover}_{2,1}\}$.*

4.4 Semantics for Relaxed Geo-Spatial Operators

As already introduced, the relaxation predicate rp, for both selection and join operators, has the aim to discard features which do not satisfy certain properties. In the following, we present different semantics for the selection (Section 4.3.1) and join (Section 4.3.2) operators, based on different definitions of the relaxation predicate rp, addressing different issues in query relaxation.

- *Best Fit semantics (BF)*: the semantics is relaxed by returning features which better satisfy the condition. Under the *Best Fit (BF) semantics* we introduce the smallest amount of relaxation in order to return a non-empty answer (if the input map is not), which is closest to the original request.
- *Threshold semantics (Thr)*: the query is executed by considering a threshold value that represents the minimum accepted similarity of the topological query relation with respect to the relations existing between the considered features. In this case an empty map could be returned if the similarity between the topological relations,

Semantics	type of condition	type of relaxation
Best Fit (BF)	well-defined and definable	not directly controlled by the system, but depending on the data at hand
Threshold (Thr)	well-defined and definable	directly controlled by the system using a threshold value, depending on the topological query relation
Neares Neighbor (NN)	definable	not directly controlled by the system, but depending on the data at hand

Table 4.2: Semantics comparisons.

existing among the considered objects, and the topological query relation is lower than the specified threshold. We decided to consider a system threshold and not a user-specified threshold (thus making relaxation implicit) since the user, in order to specify a reasonable value, should know details concerning the topological similarity, and this cannot be assumed in general. Different thresholds may generate different result sets.

- *Nearest Neighbor semantics (NN)*: if the query condition is definable, the query relation is replaced by the most similar ones, defined for the considered feature type and object dimensions. Nearest Neighbor semantics can however return an empty answer when no object in the dataset satisfies the replacing set of topological relationships. Notice that on well-defined topological conditions, the Nearest Neighbor semantics degenerates to a non-relaxed semantics, searching for the exact topological query relation. The Nearest Neighbor semantics can also be considered as a particular case of the Threshold semantics where the threshold value considered by the system is, for each considered query, the maximum similarity value.

4.4.1 Semantics for Relaxed Selection Operator

In the following, we present and discuss the semantics of the relaxed selection operator for well-defined and definable topological selection conditions, by providing for each case a formal definition of the corresponding relaxation predicate rp.

4.4.1.1 Semantics for well-defined conditions

Best Fit semantics. Under the *Best Fit (BF)* semantics, we introduce the smallest amount of relaxation in order to return a non-empty answer (if the input map is not), which is closest to the original request. Informally, the basic idea is to return all the features which *best fit* the user request. Such features are those for which the similarity

between the query relation and the topological relation existing between the features and the query object O is the highest. Of course, if features exist satisfying the query condition, no relaxation is applied. The relaxation predicate, under the *Best Fit (BF)*, semantics is therefore defined as follows:

Definition 24 (*Best Fit relaxation predicate for well-defined conditions*) Let $ft\theta^q O$ be a well-defined selection condition. The relaxation predicate, under the $Best\,Fit\,(BF)$, semantics is defined as follows:

$rp(M, ft\theta^q O, \mu) \equiv \nexists f' \in M : f' \in ext(ft, M) \wedge\ f'\theta^m O$ holds in $M \wedge\ ts(\theta^m, \theta^q) > \mu$. □

Example 15 *Suppose to execute selections Q_1, Q_2 and Q_3, presented in Section 4.2 which are based on well-defined selection conditions. Under the BF semantics, queries Q_1 and Q_2 return the same result than a non-relaxed selection operator, since there exist objects in the map satisfying the query condition (for them similarity value is 1). This is no true for query Q_3, which generates an empty result set under the non-relaxed semantics, since no main rivers are* `in` *Liguria, under the BF semantics generates an approximated result, corresponding to the main rivers* `coveredby` *Liguria, i.e., only river $F10$ is returned. Indeed, based on the similarity values reported in Tables 3.4, 3.5,* `coveredby` *is the most similar relation to* `in` *and features exist satisfying it.* □

Threshold semantics. The peculiarity of the *Threshold (Thr) semantics* is to enable the system to directly control the quality of the relaxation introduced by the selection operator, using a threshold value ρ, between 0 and 1.

All features involved in a topological relation whose similarity w.r.t. the topological query relation is equal to or more than ρ are returned to the user. Thus, relaxation is always applied; of course, different thresholds may generate different result sets.

We decided to consider a system threshold and not a user-specified threshold (thus making relaxation implicit) since the user, in order to specify a reasonable value, should know details concerning the topological similarity, and this cannot be assumed in general.

Notice that an empty map could be returned if the similarity between the topological relations, existing in the map, and the topological query relation is lower than ρ. Let $\rho \in$[0..1] be a system threshold. The relaxation predicate under the Threshold semantics is therefore defined as follows:

Definition 25 (*Threshold relaxation predicate for well-defined conditions*) Let $\rho \in$ [0..1] be a system threshold. The relaxation predicate under the threshold semantics is defined as follows:

$$rp(M, ft\theta^q O, \mu) \equiv (\mu >= \rho)$$

□

Notice that the system threshold can be chosen by the system depending on the query topological relation at hand. For example, a good choice for evaluating condition $ft\theta^q O$ could be that of taking as threshold the average similarity between $\theta^q_{d(ft),d(O)}$ and the other topological relations in $\mathcal{T}_{d(ft),d(O)}$ (excluding θ^q).

Example 16 *Consider the example presented in Section 4.2. Suppose to apply the Threshold semantics, taking* 0.556 *as threshold value (average similarity between* $\mathtt{o}_{2,2}$ *and the other topological relations in* $\mathcal{T}_{2,2}$*, excluding* $\mathtt{o}_{2,2}$*), in executing query* Q_1*, where the query condition is* $\mathtt{o}_{2,2}$*. In this case, based on the similarity values reported in Tables 3.4, 3.5, objects satisfying either* $\mathtt{o}_{2,2}$*,* $\mathtt{i}_{2,2}$*,* $\mathtt{b}_{2,2}$*,* $\mathtt{c}_{2,2}$*, or* $\mathtt{v}_{2,2}$*, with respect to the query object, are returned as result, since the similarity value between* $\mathtt{o}_{2,2}$ *and each of these relations is greater than the threshold. Thus, in this case, features* $F5$*,*$F2$*,*$F3$*,*$F9$*,*$F11$*,*$F6$ *and* $F10$ *are returned.* □

4.4.1.2 Semantics for definable conditions

Similarly to well-defined selection conditions, different definitions for predicate rp can be provided. Semantics for definable selection conditions can be defined based on the semantics for well-defined selection conditions specified using relations in $\mathcal{F}_s$. We provide three semantics for definable conditions: BF, Thr and also NN.

Best Fit semantics. In case of Best Fit semantics, since only the results which better approximate the requested condition should be returned, we decide to return those features whose similarity with respect to the request condition is the highest. This behavior can be achieved by the following relaxation predicate.

Definition 26 (*Best Fit relaxation predicate for definable conditions*) Let $C \in \mathcal{SC}_{MS_a}$. Then,

$rp(M, C, \mu) \equiv \nexists f' \in M : f' \in ext(ft, M) \wedge f'\theta^m O$ holds in $M \wedge\ ts(\theta^m, \mathcal{F}_s(C)) > \mu$. □

Best Fit semantics for definable conditions thus corresponds to execute the set of corresponding selection conditions based on the well-defined relations in $\mathcal{F}_s(C)$, take the union, and remove from them those features whose maximum similarity with respect to all the well-defined relations in $\mathcal{F}_s(C)$ is not the highest.

Threshold semantics. For Threshold semantics, it seems reasonable to execute a selection query for each condition based on the well-defined relations in $\mathcal{F}_s$ and take the union of the results. Indeed, all the partial results are equally good with respect to the query. Thus, the relaxation predicate can be defined as follows:

Definition 27 (*Threshold relaxation predicate for definable conditions*) Let $C \in \mathcal{SC}_{MS_a}$. Then,

$rp(M, C, \mu) \equiv (\mu >= \rho)$. □

Nearest Neighbor semantics. Also in this case, it seems reasonable to execute a selection query for each condition based on the well-defined relations in $\mathcal{F}_s(C)$ and take the union of the results. Thus, the relaxation predicate can be defined as follows:

Definition 28 (*Nearest Neighbor relaxation predicate for definable conditions*) Let $C \in \mathcal{SC}_{MS_a}$. Then,

$rp(M, C, \mu) \equiv (\mu = 1)$. □

We present now an example for all the semantics introduced above.

Example 17 *Consider the maps presented in Section 4.2 and query Q_4. In order to rewrite query Q_4, we first determine the closest feature type dimension for which* `overlap` *is defined and, in case more than one exists, we take the highest one.*

Such dimension is $d = 2$. Now, considering Tables 3.4, 3.5, we find that $\mathcal{F}_s(C)$={ $\texttt{cross}_{1,2}$ } *is the only topological relation which is closest to $\texttt{o}_{2,2}$.*

Query Q_4 is, thus, rewritten into $\sigma_{<Rl\ \texttt{cross}\ Liguria}(M_{Rl})$. Under the BF semantics, the railways returned are those satisfying $\texttt{r}_{1,2}$, i.e., R6, R8, etc... On the other side, under the Thr semantics, considering as threshold value 0.815, we get as results the features whose similarity with respect to $\mathcal{F}_s(C)$ is equal or higher than 0.815. Based on Tables 3.4, 3.5, we return all railways which satisfy relations `cross` *or* `coveredby` *w.r.t. Liguria. Finally, under the NN semantics, we consider the rivers that satisfy the closest relation to* `overlap`*, that is* `cross`. □

4.4.2 Semantics for Relaxed Join Operator

The approach for join operator is similar to those presented for selection operator.

4.4.2.1 Semantics for well-defined conditions

Best Fit semantics. In this case, the basic idea is to return all pairs of features which satisfy a topological relation, such that the similarity between the topological relation existing among them and that specified in the query is the highest.

Definition 29 (*Best Fit relaxation predicate for well-defined conditions*) Let $ft_a \theta^q ft_b$ be a join condition. The relaxation predicate is defined as follows:

$$rp(M_a, M_b, ft_a \theta^q ft_b, \mu) \equiv \nexists f'_a \in M_a, \nexists f'_b \in M_b : f'_a \in ext(ft_a, M_a) \wedge f'_b \in ext(ft_b, M_b) \wedge f'_a \theta^m f'_b \text{ holds} \wedge ts(\theta^m, \theta^q) > \mu. \square$$

Threshold semantics. Under the Threshold semantics, we return all pairs of features that satisfy a topological relation similar to the topological query relation more than or equal to the given threshold.

Definition 30 (*Threshold relaxation predicate for well-defined conditions*) Let $ft_a \theta^q ft_b$ be a join condition, and let $\rho \in [0..1]$ be a system threshold. The relaxation predicate under the *Threshold* (Thr) semantics is defined as follows:

$$rp(M_a, M_b, ft_a \theta^q ft_b, \mu) \equiv (\mu >= \rho). \square$$

4.4.2.2 Semantics for definable conditions

Similarly to well-defined join conditions, different definitions for predicate rp can be provided for the definition of definable join condition semantics.

Best Fit semantics. Best Fit semantics correspond to execute the set of corresponding join conditions based on the well-defined relations in $\mathcal{F}_j$, take the union discarding those pairs of features whose maximum similarity with respect to all the well-defined relations in $\mathcal{F}_j$ is not the highest.

Definition 31 (*Best Fit relaxation predicate for definable join condition*) Let $C \in \mathcal{JC}_{MS_a, MS_b}$

$$\begin{aligned} &rp(M_a, M_b, C, \mu) \equiv \\ &\quad \nexists f'_a \in M_a \wedge \nexists f'_b \in M_b : f'_a \in ext(ft_a, M_a) \wedge f'_b \in ext(ft_b, M_b) \wedge \\ &\quad f'_a \theta^m f'_b \text{ holds} \wedge \\ &\quad (f'_a, f'_b) \in \bigcup_{i=1,..,n} \bowtie_{<ft_a \theta_i^q ft_b>} (M_a, M_b) \wedge \\ &\quad ts(\theta^m, \mathcal{F}_j) > \mu. \square \end{aligned}$$

Threshold semantics. Definition 32 (*Threshold relaxation predicate for definable join condition*) Threshold semantics correspond to execute a join query for each condition based on the well-defined relations in $\mathcal{F}_j$ and simply take the union of the results, since, in this case all the partial results are equally good with respect to the query. Assuming the threshold $\rho \in [0, 1]$, the relaxation predicate can be defined as follows: Let $C \in \mathcal{JC}_{MS_a,MS_b}$

$rp(M_a, M_b, C, \mu) \equiv (\mu >= \rho).\square$

Nearest Neighbor semantics. Similarly to Threshold semantics, also the Nearest Neighbor semantics corresponds to executing the set of join conditions based on the well-defined relations in $\mathcal{F}_j$ and take the union.

Definition 33 (*Neares Neighbor relaxation predicate for definable join condition*) Let $C \in \mathcal{JC}_{MS_a,MS_b}$

$rp(M_a, M_b, C, \mu) \equiv (\mu = ts(\theta^m, \mathcal{F}_j(C)).\square$

We present now an example for all the semantics introduced for definable conditions.

Example 18 *Consider the maps presented in the Example 4.2 and the query* $M_{Mriv} \bowtie_{<Rv\ \texttt{cross}\ Pro>} M_{Pr}$ *based on definable join conditions. In order to rewrite query we first determine the closest pair of feature type dimensions for which* `cross` *is defined and, in case more than one exists, we take the pair with their sum is the highest. If more pairs have the highest sum value, we randomly take one of them. The pairs of dimensions for which* `cross` *is defined are (1,1), (2,1) and (1,2); by considering the sum of the two dimension of each pairs, we have: 1+1=2; 2+1=3, 1+2=3. We therefore choose either (2,1) or (1,2). Assume to consider (2,1). Now, considering Tables 3.4, 3.5, we find that* `overlap` *is the only topological relation which is closest to* `cross` *and which is defined for pairs of regions. Thus, the query is rewritten into* $M_{Mriv} \bowtie_{<Rv\ \texttt{overlap}\ Pro>} M_{Pr}$. *Under the BF semantics, the returned pairs are those satisfying* $\mathtt{o}_{2,2}$, *that are (F5,SV), (F5, AL), (F2,SV), (F2, AL), (F3,SV), (F3, AL), (F9,GE), (F9, AL), (F11,GE), (F11,PC), (F6,GE), (F6,PC). On the other side, under the Thr semantics, considering as threshold value 0.815, the returned pairs are those returned for the BF semantics, plus (F13,GE), (F10, IM), (F10, SV), since they satisfy one of the relations (i.e.,* $o_{2,2}$, $b_{2,2}$, $v_{2,2}$, $c_{2,2}$) *similar to* $o_{2,2}$ *more or equal than 0.815 (see again Tables 3.4, 3.5. Finally, under the NN semantics we consider the rivers that satisfy the closest relation to* $o_{2,2}$, *that is* $o_{2,2}$ *itself. Thus, we return the same rivers returned by the Best Fit semantics.* $\square$

4.5 Query Processing for Relaxed Geo-Spatial Operators

In order to provide an efficient algorithm for executing approximated topological selections, we assume, as usual in the geo-spatial context, that instances of each feature type ft in a map M are indexed by an R-Tree [Gut84] or one of its variants, R^+-trees [SRF87] and R^*-trees [BKSS90], for guaranteeing a fast access. An R-tree entry pointing to an intermediate node (a leaf node, an object) is called *intermediate entry* (*leaf entry, object entry*) in the following. Moverover, Table 4.3 summarizes the R-tree notation we use in the following sections.

Symbol	Description
O, $d(O)$	query object, dimension of the query object
ft_i, $d(ft_i)$	feature type requested by the query belonging to map M_i, dimension of the feature type
θ^q	topological query relation
$R_i, E(R_i)$	R-tree representing map M_i, the set representing all the entries of R_i
o_i, $o_i.r$	object indexed by R_i MBR of an object o_i
e_i, $e_i.r$	R-tree R_i entry, MBR associated with entry e_i
$e_i.o$	object pointed by e_i, if e_i is an object entry
$e_i.n$	node pointed by e, if e is not an object entry
$e_i.do$	set of objects indexed by the subtree rooted by e_i
$e_i.top(e_j)$	set of topological relations between objects in $e_i.do$ and the objects in $e_j.do$, $e_j \in E(R_j) \cup O$

Table 4.3: Frequently used symbols.

4.5.1 The basic idea

In designing the algorithms for executing the topological relaxed selection operator, we use a branch and bound approach. Similarly to what has been done in [HS99] and [ZPZL05], interesting subtrees are visited as early as possible and non-interesting subtrees (those which cannot provide any new interesting result) are early discarded, based on some pruning condition.

The visit of a subtree can be avoided if and only if the objects pointed by its leaves do not belong to the result. To guarantee efficiency, such property should be checked locally at the node under consideration, using some key value. The key is also used in determining the ordering upon which nodes are visited, with the aim of visiting first the nodes which most probably will generate some suitable results.

The relaxed topological selection (join) operator returns the objects (pairs of objects) with the highest similarity between the topological relation existing among them and the query

object (among the objects of the pairs) and the topological relation specified in the query. Thus, the key value associated with each R-tree entry e (that for join is the entry of the first R-tree) should represent the range of similarity values for objects in $e.do$ with respect to the query condition $ft\theta^q O$ ($ft_a \theta^q ft_b$). In order to determine whether e should be accessed (and its subtree visited) or it can be discarded without loosing any result, such interval is compared with a *pruning value*. Such value corresponds to the maximum lower bound for similarity values of the result set, based on entries and objects already visited.

The topological similarity interval represents the maximum and minimum similarity values between the set of possible topological relations between objects contained in the subtree rooted by the considered entry e_i of the first R-tree and the objects contained in the subtree rooted by the considered entry e_j of the second R-tree (for join operator) or the object O (for selection operator).

Definition 34 (*Topological similarity interval for selection (join)*) Let e_a be an entry of R_a. The topological similarity interval associated with e_a w.r.t a relation θ^q and a query object O (an entry e_b of R_b) is defined as follows:

$tsi(e_a, \theta^q, O) = [\ min\{ts(\theta^q, \overline{\theta}) | \overline{\theta} \in e_a.top(O)\},\ max\{ts(\theta^q, \overline{\theta}) | \overline{\theta} \in e_a.top(O)\}\ ]$

$(tsi(e_a, \theta^q, e_b) = [\ min\{ts(\theta^q, \overline{\theta}) | \overline{\theta} \in e_a.top(e_b)\},\ max\{ts(\theta^q, \overline{\theta}) | \overline{\theta} \in e_a.top(e_b)\}\ ]).$

We denote with $tsi(e_a, \theta^q, O).i$ ($tsi(e_a, \theta^q, e_b).i$), $i = 1, 2$, the left and the right end points of the interval. □

Computing $tsi(e_a, \theta^q, O)$ ($tsi(e_a, \theta^q, e_b)$) requires accessing all objects in $e_a.do$ ($e_a.do$ and $e_b.do$), which is obviously an expensive operation. In order to make such computation local at each node, we compute an approximated topological similarity interval relying on the notion of *compatibility* for topological relations in an R-tree, presented in [PTSE95] for pairs of regions, here extended to any object dimension. The basic idea behind compatibility is that, given $o_a \in e_a.do$ ($o_a \in e_a.do$ and $o_b \in e_b.do$) if $o_a\ \theta\ O$ ($o_a\ \theta\ o_b$) holds, the topological relation between $o_a.r$ and O ($o_b.r$), as well as between ancestor MBRs of $o_a.r$ and O ($o_b.r$), cannot be arbitrary but must be *compatible* with θ.

Definition 35 (*Compatibility for selection operator*) Suppose that $e_a.r\ \theta\ O.r$ holds. θ is compatible with θ^q if it may exist $o_a \in e_a.do$ such that $o_a\ \theta^q\ O$ holds. If e_a is a leaf entry, the set of topological relations in $\mathcal{T}_{d(ft_a),d(O)}\ \theta$ is compatible with is denoted by $cl(\theta, d(ft_a), d(O))$; if e_a is an intermediate entry, it is denoted by $(ci(\theta, d(ft_a), d(O)))$. □

For the join operator the definition is the same, it is sufficient to substitute the query object O, with a generic entry e_b belonging to the second R-tree considered in the join; and object dimension $d(O)$ with the dimension of a generic feature $d(f_b)$ belonging to the leaf e_b of the second R-tree considered.

Table 4.4 summarizes selection compatibility rules for leaf and intermediate entries, for objects of any dimension. The first column of the table contains the relation θ satisfied by $e_a.r$ and $O.r$; the second column points out the dimension of $O.r$, since both $c()$ and $co()$ may change when MBRs degenerates to points; the fourth and the fifth columns contain $c(\theta, d(ft), d(O))$ and $co(\theta, d(ft), d(O))$, respectively, for any pair of object dimension $(d(ft), d(O))$, pointed out in the third column.

Compatibility rules for join operator are shown in Table 4.5. With respect to Table 4.4, the second column points out the dimension of $(d(e_a.r), d(e_b.r))$, since both $ci()$ and $cl()$ may change when MBRs degenerates to points.

Notice that, in general, $cl(\theta, d(ft), d(O)) \subseteq ci(\theta, d, d(O))$. Additionally, the column for leaves $cl()$ is equal in both tables, while the $ci()$ column changes. This is due to the fact that for selection, the MBR of the query object O does not change during the visit of the R-tree; on the other side, for join the MBRs of the entries considered in each step of the visit change, thus affecting computation of the compatibility of intermediate entries.

Example 19 *Suppose we are executing the selection query* $\sigma_{<Rivers\ \texttt{overlap}\ O>}(M_{MRiv})$*, and* $d(Rv) = d(O) = 2$ *(thus, we deal with pairs of regions). Suppose that during the visit of the R-tree indexing* M_{MRiv}*, we consider an entry* e_{Rv} *such that the relation* $e_{Rv}.r$ `disjoint` $O.r$ *holds. This means that objects contained in the subtree rooted by* e_{Rv} *are either* `disjoint` *with respect to* O*. Suppose we are executing the join query:* $\bowtie_{<Rv\ \texttt{overlap}\ Pro>}(M_{MRiv}, M_{Pr})$*, and* $d(Rv) = d(Pro) = 2$*, thus the objects of both R-trees are regions. Suppose during the visit of the R-trees indexing* M_{MRiv} *and* M_{Pr} *we find that the relation* $e_{Rv}.r$ `touch` $e_{Pro}.r$ *holds. This means that objects contained in* e_{Rv} *and in* e_{Pro} *are either* `disjoint` *or* `touch` *with respect to* O*.* □

By using the notion of compatibility, we can therefore infer the set of relations that objects in $e_a.do$ may satisfy with respect to O ($e_b.do$). Such set is of course a superset of $e.top(O)$ as pointed out by the following proposition.

Proposition 5 *Suppose that* $e_a.r\ \theta\ O.r$ *(*$e_a.r\ \theta\ e_b.r$*) holds. Then:*

$e_a.top(O) \subseteq cl(\theta, d(ft_a), d(O)) \subseteq ci(\theta, d(ft_a), d(O))$

($e_a.top(e_b) \subseteq cl(\theta, d(ft_a), d(ft_b)) \subseteq ci(\theta, d(ft_a), d(ft_b))$*)*

□

We can now compute $tsi(e_a, \theta^q, O)$ ($tsi(e_a, \theta^q, e_b)$) in an approximated way (denoted by tsi^a) as follows:

Definition 36 *(Approximated similarity interval)*

$$tsi^a(e_a, \theta^q, O) = [\ min\{ts(\theta^q, \overline{\theta}) | e_a.r\ \theta\ O.r \wedge \overline{\theta} \in c(\theta, d(ft_a), d(O))\},\\ max\{ts(\theta^q, \overline{\theta}) | e_a.r\ \theta\ O.r \wedge \overline{\theta} \in c(\theta, d(ft_a), d(O))\}\ ]$$

$$(tsi^a(e_b, \theta^q, e_b) = [\ min\{ts(\theta^q, \overline{\theta}) | e_a.r\ \theta\ e_b.r \wedge \overline{\theta} \in c(\theta, d(ft_a), d(ft_b))\},\\ max\{ts(\theta^q, \overline{\theta}) | e_a.r\ \theta\ e_b.r \wedge \overline{\theta} \in c(\theta, d(ft_a), d(ft_b))\}\ ])$$

where c is either $cl()$ or $ci()$ depending on whether e_a (e_a, e_b) is a leaf (leaves) or intermediate entry (entries) . □

In order to compare key values, we then define the following ordering between intervals.

Definition 37 *(Order relation between similarity intervals)*

$tsi(e_a, \theta^q, O) < tsi(e_b, \theta^q, O)$ if and only if one of the following conditions hold:

- $tsi(e_a, \theta^q, O).2 < tsi(e_b, \theta^q, O).2$
- $tsi(e_a, \theta^q, O).2 = tsi(e_b, \theta^q, O).2$ and $tsi(e_a, \theta^q, O).1 < tsi(e_b, \theta^q, O).1$. □

The difference for join operator is to consider instead of the query object O two R-tree entries e_{ab} and e_{bb} belonging to the two R-trees considered in the join.

4.5.2 The Query Processing Algorithms

The skeleton of all the presented algorithms, both for selection and join, is similar. In the following, we discuss the algorithm for Best Fit semantics for selection and join operators (Algorithm 4, Algorithm 7), then we discuss the differences in supporting the other semantics.

We visit the R-tree R_a (or both the R-trees R_a and R_b) starting from the root (roots) and we maintain a priority queue H of entries (pairs of entries) and objects (pairs of objects), that may generate some results, together with their tsi^a; H is ordered as discussed in Definition 37. The pruning condition pc is the maximum lower bound for the similarity

values of the result set, computed based on entries and objects already visited. The queue H is such that at any instant of time it contains only entries (pairs of entries) and objects (pairs of objects) whose tsi^a has a non-empty intersection with the pruning interval $[pc, 1]$.

Then, we get the maximum element *elem* (pairs of elements $(elem_a, elem_b)$) of H, by function GETMAX. The result is progressively generated in an ordered way. All algorithms starts from the root of the tree (of the two trees), which is put into H, and pc is initialized to 0. Then, for each element of the queue, starting from the maximum one, obtained by function GETMAX, three cases may arise:

- If *elem* is an object (pairs of objects), it will certainly belong to the result and it can be returned as output (indeed, since H is ordered, any entry or object following the object at hand in H will generate lower similarity values with respect to the query).
- If *elem* is an intermediate entry (pairs of intermediate entries), the entries of the pointed node (the entries of both the pointed nodes) are accessed. For the join operator there is a third case if *elem* is a intermediate entry and the other entry is a leaf (or viceversa). In this case, only the entries of the intermediate node are accessed and the pairs enqueued are formed by the leaf and the entries of intermediate node. For each accessed entry e_a (pair of entries (e_a, e_b)), we compute tsi^a, based on Table 4.4 for selection (on Table 4.5 for join) and, if based on the pruning condition pc, we determine whether the sub-tree rooted by $e_a.i$ (or $e_a.i$, $e_b.i$) should be visited since it might generate further results. This happens when $tsi^a(e_a, \theta^q, O).2$ ($tsi^a(e_a, \theta^q, e_b).2$) is greater than or equal to pc. In this case, e_a ((e_a, e_b)) is inserted in H in order to be visited, otherwise it is discarded. If also $tsi^a(e_a, \theta^q, O).1$ ($tsi^a(e_a, \theta^q, e_b).1$) is greater than or equal to pc, this means that $tsi^a(e_a, \theta^q, e_b).1$ can be taken as a new lower bound for similarity values of the result set. Thus, pc is updated and all the elements in H that cannot produce further results, based on the new pc, are removed using function CLEARQUEUE (this happens for all queue elements e such that $tsi^a(e_a, \theta^q, O).1 < pc$ ($tsi^a(e_a, \theta^q, e_b).1 < pc$)).
- Finally, if *elem* is a leaf entry (pair of leaves), the entries of the pointed leaf (leaves) are accessed. For each of them, we compute tsi^a, based on Table 4.4 (Table 4.5) and, if based on the pruning condition the entry (pair of entries) may produce further results, we perform the following steps: we access the corresponding pointed object (pair of objects), we determine the topological relation between it and O (between the two objects), we compute the topological similarity and, if it is greater than pc, we insert it (the pair of objects) in H since it (they) may belong to the result. In this case, the computed topological similarity can be taken as a new lower bound for similarity values of the result set. Thus, pc is updated and all the elements in H that cannot produce further results based on the new pc are removed.

Since the number of possible similarity values is fixed, the number of possible priority intervals is fixed as well. This consideration allows one to use an implementation of H which guarantees constant time for insertion and GETMAX operations. Additionally, each entry or object is inserted in the queue at most once (and therefore it can be removed just once from it). This means that the overhead in using the priority queue is bound by the number of entries and objects visited in the R-tree.

In Section 4.5.3, we provide algorithms for well-defined selection conditions, together with a discussion concerning how such algorithms can be used for definable selection conditions. While the correspondent algorithms for well-defined join conditions and the correspondent approach for definable join condition are proposed in Section 4.5.4.

4.5.3 Query Processing for Relaxed Selection Operator

In the following, we present the query processing algorithms for the relaxed selection operator, inpresence of both well-defined and definable conditions. For each semantics, the differences with respect to the skeleton presented in Section 4.5.2 will be discussed.

4.5.3.1 Query processing for Well-defined Selection Conditions

In Section 4.4 we have formally defined two semantics for well-defined selection conditions, Best Fit and Threshold, and in the following, we briefly introduce and then we discuss their algorithms.

As already discussed the skeleton of both algorithms is similar. The only difference between Best Fit and Threshold algorithms is in the initialization and in the updating of the pruning condition pc, as explained in the following.

Best Fit semantics. Since the skeleton is based on the Best Fit semantics, there are no differences to highlight. We remember that, in this case the pruning condition is the lowest bound for the similarity values of best fit solutions, based on entries and objects visited so far. We then update the pruning interval if, from tsi^a of the current entry/object e, we get an higher pruning interval. This happens if $tsi^a(e, \theta^q, O).1$ is greater than pc (in this case pc is updated). The query processing algorithm is presented in Algorithm 4.

Threshold semantics. The query processing algorithm for Threshold semantics is obtained from Algorithm 4 by taking into account that: (i) the pruning condition pc in this case is fixed and corresponds to the threshold value; (ii) pc never changes since it does not depend on the entries/objects visited so far. The query processing algorithm is presented in Algorithm 5.

Algorithm 4 Algorithms for Selection: BF semantics.

INPUT
Feature type ft
Query object O
Query relation θ^q
R-tree R over ft instances in map M
Ensure: Objects in $\sigma_{ft\theta^q O}$ under the Best Fit semantics
METHOD
PriorityQueue H:= empty
H.insert($R.root$,0,0)
pc:=0
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem$ is an intermediate entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,θ^q,O)
      if $simmin >= pc$ **then**
        $pc := simMin$
        clearQueue(H,pc)
      end if
      if $simmax >= pc$ **then**
        H.insert(e,$simmin$,$simmax$)
      end if
    end for
  else if $elem$ is a leaf entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,θ^q,O)
      if $simmax >= pc$ **then**
        determine θ^m between $e.o$ and O
        $RankVal$:=ts($\theta^m_{d(ft),d(O)}$, $\theta^q_{d(ft),d(O)}$)
        if $RankVal > pc$ **then**
          $pc := RankVal$
          clearQueue(H,pc)
          H.insert($e.o$,$RankVal$,$RankVal$)
        end if
        if $RankVal = pc$ **then**
          H.insert($e.o$,$RankVal$,$RankVal$)
        end if
      end if
    end for
  else
    output $elem$, $RankVal$
  end if
end while

Example 20 *Consider the R-tree in Figure 4.2 and the query object O, as pointed out in the figure. Suppose that objects indexed by the R-tree and O are regions and assume, for the sake of simplicity, that each MBR in leaf nodes contains just one object coinciding with the MBR itself, also for O. Only the objects coinciding with leaf MBRs $R13$ and $R14$ are finally checked. Let O_{R13} be the object contained in $R13$ and O_{R14} the object contained in $R14$. None of them is contained in O. However, O_{R13}* `overlap` *O while O_{R14} is* `coveredby` *O.*

Now suppose you want to determine, under the Best Fit semantics, the objects `in` *O. Based on the similarity value, only O_2 is returned. In the following, we discuss in detail the steps*

Algorithm 5 Algorithms for Selection: Thr semantics.

INPUT
Feature type ft
Query object O
Query relation θ^q
R-tree R over ft instances in map M
Threshold ρ
Ensure: Objects in $\sigma_{ft\theta^q O}$ under the Threshold semantics
METHOD
PriorityQueue H:= empty
H.insert($R.root$,0,0)
pc:=ρ
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem$ is a intermediate or leaf entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,θ^q,O)
      if $simmax >= pc$ **then**
        H.insert(e,$simmin$,$simmax$)
      end if
    end for
  else if $elem$ is a leaf entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,θ^q,O)
      if $simmax >= pc$ **then**
        determine θ^m between $e.o$ and O
        $RankVal$:=ts($\theta^m_{d(ft),d(O)}, \theta^q_{d(ft),d(O)}$)
        if $RankVal >= pc$ **then**
          H.insert($e.o$,$RankVal$,$RankVal$)
        end if
      end if
    end for
  else
    output $elem$, $RankVal$
  end if
end while

of the Best Fit execution.

Step 1: *We initialize H with the root and with the tsi=[0,0] and the pruning condition with 0.*

Step 2: *We extract the first element of H, and we explore its entries: $R1$, $R2$. We start by analyzing $R1$. $R1$* `c` *O holds; by using Table 4.4 and considering ci(), we obtain the set of relations compatible with* `c`*, corresponding to the set* {`d`,`t`,`i`,`e`,`c`, `v`,`b`,`o`}*. Then, we compute the similarity interval [simmin,simmax]=[0.463,1], since ts(i, d)= 0.463 and ts(i, i)= 1. The pruning condition is updated, since pc = 0.463>0, and the queue is not cleared since it is empty. Finally, the element is inserted in the queue $H = [(R1, 0.463, 1)]$.*

Step 3: *Now $R2$ is analyzed. $R2$* `t` *O holds, by using Table 4.4 and considering ci(), we obtain the set of relations compatible with* `t`*, corresponding to the set* {`d`,`t`}*. Then,*

we compute the similarity interval [simmin,simmax]=[0.463,0.519], since ts(i, d)= 0.463, and the ts(i, t)= 0.519. The pruning condition is not updated since simmin=0.463=pc and the queue is not cleared. R2 is then inserted in the queue in an ordered way, after R1, since value for simmax is 0.519, which is lower than the simmin value for R1, which is 1. Thus, $H = [(R1, 0.463, 1), (R2, 0.463, 0.519)]$.

Step 4: *Now R1 is dequeued and its entries are analyzed. R3* `d` *O holds, by using the Table 4.4 considering ci(), we obtain the set of relations compatible with* `d`*, the set is* {`d`}*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.463], since ts(i, d)= 0.463. The pruning condition is not updated since simmin=0.463=pc, and the queue is not cleared. R3 is inserted in the queue in an ordered way. R3 is the last element since it has the lowest value for simmin,* $H = [(R2, 0.463, 0.519), (R3, 0.463, 0.463)]$.

Step 5: *Another entry of R1 is analized. R4* `t` *O holds, by using the Table 4.4 considering ci(), we obtain the set of relations compatible with* `t`*, the set is* {`d`,`t`}*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.519], since ts(i, d)= 0.463, ts(i, t)= 0.519. The pruning condition is not updated since simmin=0.463=pc, and the queue is not cleared. R4 is inserted in the queue in an ordered way. R4 is positioned in a random way before or after R2, since their similarity intervals are equal.* $H = [(R2, 0.463, 0.519), (R4, 0.463, 0.519), (R3, 0.463, 0.463)]$.

Step 6: *The last entry of R1 is analized. R5* `o` *O holds, by using the Table 4.4 considering ci(), we obtain the set of relations compatible with* `o`*, the set is* {`d`,`t`,`i`,`b`,`o`}*. Then we compute the similarity interval [simmin,simmax]=[0.463,1], since ts(i, d)= 0.463, ts(i, i)= 1. The pruning condition is not updated since simmin=0.463=pc, and the queue is not cleared. The R5 is inserted in the queue in an ordered way. R5 is in the first position since their similarity its simmax=1, is the highest one, since simmin is equal for all elements.* $H = [(R5, 0.463, 1), (R2, 0.463, 0.519), (R4, 0.463, 0.519), (R3, 0.463, 0.463)]$.

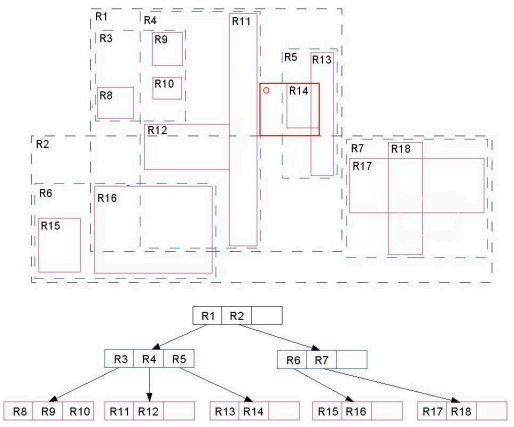
Step 7: *R5 is dequeued and its entries are analized. The first is R13* `o` *O holds, using the Table 4.4 considering cl(), since R13 is an entry leaf, we obtain the set of relations compatible with* `o`*, the set is* {`d`,`t`,`o`}*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.603], since ts(i, d)= 0.463, ts(i, o)= 0.603. The pruning condition is not updated since simmin=0.463=pc, and the queue is not cleared. Since simmax=0.463>=pc we consider the object* O_{R13}*;* O_{R13} `o` *O holds, thus RankVal(*O_{R13}*) =ts(i,o)= 0.603, thus [simmin,simmax]=[0.603,0.603]. The value of pc is updated to 0.603, and H is cleared eliminating the elements which simmax is lower than 0.603, H=[]. The* O_1 *is inserted in the queue in an ordered way. R5 is in the first position since their similarity its simmax=0.863, is the highest one, since simmin is equal for all elements.* $H = [(R13, 0.603, 0.603)]$*, pc:=0.603.*

Step 8: *The first last entry of R5 is R14. R14* `b` *O holds, using the Table 4.4 considering cl(), since R14 is an entry leaf, we obtain the set of relations compatible with* `b`*, the set is* {`d`,`b`,`t`,`o`}*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.889], since ts(i, d)= 0.463, ts(i, b)= 0.889. The pruning condition is not updated since simmin=0.463<pc, and the queue is not cleared. Since simmax=0.889>pc we consider the object* O_{R14}*.* O_{R14} `o` *O holds, thus RankVal(*O_{R14}*) =ts(i,b)= 0.889, thus [simmin, simmax]=[0.889,0.889]. The value of pc is updated to 0.889, and H is cleared eliminating the elements which simmax is lower than 0.889, H=[]. The* O_{R14} *is inserted in the queue in an ordered way.* $H = [(O_{R14}, 0.889, 0.889)]$*, pc:=0.889.*

Step 9: O_{R14} *is dequeued and returned as result. H is empy, the execution terminate.*

A similar execution is performed for the Threshold semantics. The difference are the following:(i) assuming $\rho = 0.6$*, the starting value for pc is 0.6 and not 0; (ii) the value of pc is never updated, but is always 0.6. Then, the computation proceeds as shown in Figure 4.2. In this case, both* O_{R13} *and* O_{R14} *are returned since their similarity with respect to the query condition is higher than 0.6.* □

Best Fit
H=[(root,0,0)] pc = 0
R1: R1 c O, [simmin,simmax]=[0.463,1], H=[(R1,0.463,1)] pc = 0.463
R2: R2 t O, [simmin,simmax]=[0.463,0.519], H=[(R1,0.463,1), (R2,0.463,0.519)] pc = 0.463
R3: R3 d O, [simmin,simmax]=[0.463,0.463], H=[(R2,0.463,0.519), (R3,0.463,0.463)] pc = 0.463
R4: R4 t O, [simmin,simmax]=[0.463,0.519], H=[(R2,0.463,0.519), (R4,0.463,0.519), (R3,0.463,0.463)] pc = 0.463
R5: R5 o O, [simmin,simmax]=[0.463,0.863], H=[(R5,0.463,0.863), (R2,0.463,0.519), (R4,0.463,0.519), (R3,0.463,0.463)] pc = 0.463
R13: R13 o O, [simmin,simmax]=[0.463,0.603] O1 o O, RankVal(O1) = 0.603, [simmin,simmax]=[0.603,0.603], H=[(R13,0.603,0.603)] pc = 0.603
R14: R14 b O, [simmin,simmax]=[0.463,0.889], O2 b O, RankVal(O2) = 0.889, [simmin,simmax]=[0.889,0.889], H=[(R14,0.889,0.889)] pc = 0.889



Threshold 0.6
H=[(root,0,0)] pc = 0.6
R1: R1 c O, [simmin,simmax]=[0.463,1], H=[(R1,0.463,1)]
R2: R2 t O, [simmin,simmax]=[0.463,0.519], H=[(R1,0.463,1)]
R3: R3 d O, [simmin,simmax]=[0.463,0.463], H=[]
R4: R4 t O, [simmin,simmax]=[0.463,0.519], H=[]
R5: R5 o O, [simmin,simmax]=[0.463,0.863], H=[(R5,0.463,0.863)]
R13:R13 o O, [simmin,simmax]=[0.463,0.603] O1 o O, RankVal(O1) = 0.603, [simmin,simmax]=[0.603,0.603], H=[(R13,0.603,0.603)]
R14: R14 b O, [simmin,simmax]=[0.463,0.889], O2 b O, RankVal(O2) = 0.889, [simmin,simmax]=[0.889,0.889], H=[(R14,0.889,0.889),(R13,0.603,0.603)]

Figure 4.2: Example R-tree; Best Fit and Threshold computations for selection operator.

4.5.3.2 Query Processing for Definable Selection Conditions

Since each definable condition is approximated by a set of well-defined conditions, processing for selection operations based on definable conditions corresponds to executing a set of selection operations based on well-defined conditions and applying a given operation to the resulting sets, according to what presented in Subsection 4.4.1.2 (union for Threshold semantics, maximum similarity for Best Fit semantics). Of course, this corresponds to visiting the R-tree once for each well-defined condition to be considered. Such an approach can be optimized by using a batch strategy, in order to execute all selection operations during just one visit of the R-tree. This is possible by introducing specific operations on topological similarity intervals, depending on the chosen semantics.

More precisely, the algorithms for Best Fit and Thresholds semantics for well-defined selection conditions can be used for executing selection conditions based on definable conditions. On the other side, the algorithm for Nearest Neighbor semantics is very similar to the ones for Threshold semantics, thus, we present together these semantics. Notice that we suppose to perform the calculation of the well-defined selection conditions as discussed and then to pass them to the algoritms.

In particular:

- the algorithms should take as input $\mathcal{F}_s(C)$;
- the computation of the topological similarity intervals (lines 14 and 25 in Algorithm 4, lines 15 and 22 in Algorithm 5, lines 15 and 22 in Algoritm 6) is substituted by the computation of n topological similarity intervals $tsi(e, \theta_i^q, O)$, $i = 1, .., n$, one for each relation in $\mathcal{F}_s(C)$, followed by an aggregate operations over such intervals, depending on the semantics.
- the computation of the ranking value (line 23 in Algorithm 4, line 25 in Algorithm 5, line 25 in Algoritm 6) is substituted in all the three algorithms by the computation of the maximum of the n topological similarity values $ts(\theta^m, \theta_i^q)$, $i = 1, .., n$.

The aggregate operations for the intervals, are the following:

Best Fit semantics. The maximum interval is computed: $(max\{tsi(e, \theta_i^q, O).1 | i = 1, ..., n\}$, $max\{tsi(e, \theta_i^q, O).2 | i = 1, ..., n\})$. See Algorithm 4.

Threshold and Neares Neighbor semantics. The union of intervals is computed: $(min\{tsi(e, \theta_i^q, O).1 | i = 1, ..., n\}$,$max\{tsi(e, \theta_i^q, O).2 | i = 1, ..., n\})$ See Algorithm 5 and Algorithm 6, respectively.

Algorithm 6 Algorithms for Selection: NN semantics.

INPUT
Feature type ft
Query object O
Well-defined query relations $\theta_1^q, \ldots, \theta_n^q$
R-tree R over ft instances in map M
Threshold ρ
Ensure: Objects in $\sigma_{ft\theta_i^q O}$ with $i = 1, \ldots, n$ under the Nearest Neighbor semantics
METHOD
PriorityQueue H:= empty
H.insert($R.root$,0,0)
pc:=1
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem$ is a intermediate or leaf entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,$\theta_1^q, \ldots, \theta_n^q$,$O$)
      if $simmax >= pc$ **then**
        H.insert(e,$simmin$,$simmax$)
      end if
    end for
  else if $elem$ is a leaf entry **then**
    for all $e \in elem.n$ **do**
      ($simmin$,$simmax$):=tsi(e,$\theta_1^q, \ldots, \theta_n^q$,$O$)
      if $simmax = pc$ **then**
        determine θ^m between $e.o$ and O
        $RankVal$:=ts($\theta^m_{d(ft),d(O)}, \theta^q_{d(ft),d(O)}$)
        if $RankVal = pc$ **then**
          H.insert($e.o$,$RankVal$,$RankVal$)
        end if
      end if
    end for
  else
    output $elem$, $RankVal$
  end if
end while

4.5.4 Query Processing for Relaxed Join Operator

For join operator, the skeleton of the processing algorithms is slightly different from the one introduced in Section 4.5. In fact, similarly to what has been done in [ZPZL05], it is necessary to distinguish between two main cases: (i) the two R-tree have the same height; (ii) the two R-tree have different heights. As a consequence, the following possible cases arise: (i) the join of a node in the first R-tree with a node in the second R-tree; (ii) the join of a node in the first R-tree with a leaf in the second R-tree; (iii) the join of a leaf in the first R-tree with a node in the second R-tree; (iii) the join of a leaf in the first R-tree with a leaf in the second R-tree. The two approaches for joining two geo-spatial indexes of different heights, $fix-at-root$ and $fix-at-leaves$ were originally proposed in [CMTV04, CMTV00]; we decide to adopt, as suggested in [ZPZL05], a technique similar to $fix-at-root$. The basic idea is to stop the downwards visit in the smallest R-tree at the leaf entry level and continue with the propagation on the highest R-tree until the leaf

entry level is reached. Then the two R-tree are visited in a synchronous way.

Similarly to selection, also for the query processing algorithms of the join operators we use the same metric for defining the pruning condition and the key. In particular, we use the topological similarity with respect to the query relation as a measure for defining value used in the pruning condition and the key. More precisely, given a query relation θ^q, we associate with each pair of entries (e_i, e_j) an interval of similarity values corresponding to the set of topological relations that objects in $e_i.do$ may satisfy with respect to at least one of the objects in $e_i.do$. The way we then use such values for computing keys and pruning conditions in the proposed algorithms, depends on the semantics we are considered as discussed in the following sections.

4.5.4.1 Query Processing for Well-defined Join Conditions

In the following, we present the query processing algorithms for the relaxed join operator, in presence of well-defined conditions. For each semantics, the differences with respect to the skeleton presented in Section 4.5.2 will be discussed.

As already discussed the skeleton of the algorithms is similar. The only difference between Best Fit and Threshold algorithms is in the inizialization and in the updating of the pruning condition *pc* as explained in the following.

Best Fit semantics. Under the Best Fit semantics, we update the pruning interval if, from *tsi* of the current pair of entries/objects (e_a, e_b), we can get an higher pruning interval. This happens if $tsi(e_a, \theta^q, e_b).1$ is greater than *pc* (in this case *pc* is updated). The query processing algorithm is presented in Algorithm 7.

Threshold semantics. For Threshold semantics, the pruning condition is still the lowest bound but it is fixed and corresponds to the threshold value. Also here, the pruning interval never changes since it does not depend on the entries/objects visited so far. The query processing algorithm is presented in Algorithm 8.

Example 21 *Consider the R-trees in Figure 4.3 and assume, for the sake of simplicity, that each MBR in leaf nodes contains just one object coinciding with the MBR itself.*

Now suppose you want to determine, under the Best Fit semantics, the pairs of objects satisfying the relation `in`*. No pairs of objects satisfy this relation. In the following, we discuss in detail the steps of the Best Fit execution.*

Step 1: *We initialize H with the roots, with the tsi=[0,0] and the pruning condition with 0.*

Step 2: *We extract the first element of H, and we explore the roots A and B. We start analyzing the entries of A with respect those of B. A have these entries: $A1$, $A2$, $A3$, $A4$, $A5$; while B have these entries: $B1$, $B2$, $B3$, $B4$, $B5$. For each pairs we calculate the similarity interval. For the sake of presentation in the following we describe in detail only the computation involving the entry $A1$ and $B1$, the others are similarly. In particular, $A1$ o $B1$, $A1$ o $B2$, $A1$ d $B3$, $A1$ d $B4$, $A1$ o $B5$ hold.*

By using the Table 4.5 considering ci(), we obtain the set of relations compatible with o*, the set is* {d,t,i,e,c, v,b,o}*. Then we compute the similarity interval [simmin,simmax]=[0.463,1], since ts(i, d)= 0.463, and the ts(i, i)= 1. Moreover, the set of relations compatible with* d*, is only* d*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.463], since ts(i, d)= 0.463. The pruning condition is updated since pc = 0.463>0, and the queue is not cleared since it is empty. Thus the ordered queue is the following* $H = [((A1, B1), 0.463, 1), ((A1, B2), 0.463, 1), ((A1, B5), 0.463, 1), ((A1, B3), 0.463, 0.463), ((A1, B4), 0.463, 0.463)]$.

Step 3: *Now* $((A1, B1), 0.463, 1)$ *is analyzed. Considering the entries of $A1$, that are $a1$, $a2$, $a3$, $a4$ and the entry of $B1$ $b1$, $b2$, $b3$, $b4$. Also here we analize in detail only the processing for $(A1, B1)$ considering in particular the entry $a1$ of $A1$, since in the others cases the computation is similar.*

The following relations hold: a1 o *b1, a1* o *b3, a1* d *b2, a1* d *b4. By using the Table 4.5 considering cl(), we obtain the set of relations compatible with* o*, the set is* {d,t, o}*, Then we compute the similarity interval [simmin,simmax]=[0.463,0.603], since ts(i, d)= 0.463, and the ts(i, o)= 0.603. The set of relations compatible with* d *that is* {d}*. Then we compute the similarity interval [simmin,simmax]=[0.463,0.463]. But since we suppose the MBR of objects are the objects itself, we can calculate the ranking value, for the relation* o *is 0.603 and for the relation* d *is 0.463. The pruning condition is updated since for overlap the ranking value is 0.603>pc, and the queue is cleared. Th ordered queue is the following,* $H = [((A1, B2), 0.463, 1), ((A1, B5), 0.463, 1), (a1, b1, 0.603, 0.603), (a1, b3, 0.603, 0.603)]$.

Step 4: *Now ((A1,B2),0.463,1) is dequeued and its entries are analyzed. Similarly to the computation performed for (A1,B1), the pairs considered are those that satisfy the relation* o*, that are (a1, b5) and (a1,b7). Also in this case the ranking value is 0.603 The pruning condition is not updated since 0.603=pc, and the queue is not cleared. The new pairs are entered in the queue. The queue is the following* $H = [((A1, B5), 0.463, 1), (a1, b1, 0.603, 0.603), (a1, b3, 0.603, 0.603), (a1, b5, 0.603, 0.603), (a1, b7, 0.603, 0.603)]$.

Step 5: *Now ((A1,B5),0.463,1) is dequeued and its entries are analyzed. Similarly to the computation performed for (A1,B1), the pairs considered are those that satisfy the relation* o*, that are (a1, b18) and (a1,b19). Also in this case the ranking value is 0.603*

The pruning condition is not updated since 0.603=pc, and the queue is not cleared. The new pairs are entered in the queue, $H = [(a1, b1, 0.603, 0.603), (a1, b3, 0.603, 0.603), (a1, b5, 0.603, 0.603), (a1, b7, 0.603, 0.603), (a1, b18, 0.603, 0.603), (a1, b19, 0.603, 0.603)]$.

Step 6: *Since all elements of the queue H are objects, they are all returned, and then the computation stop.*

No pairs of objects satisfy the query condition `i`*, the Best Fit semantics return those that better satisfy the condition. Since in this case the only two relations satisfied by the objects considered are* `d` *and* `o`*, the most similar to* `i` *is* `o`*.*

A similar execution is performed for the Threshold semantics. The difference are the following:(i) assuming ρ*= 0.6, the starting value for pc is 0.6 and not 0; (ii) the value of pc is never updated, but is always 0.6. Also in this case the pairs of objects returned are the same.* □

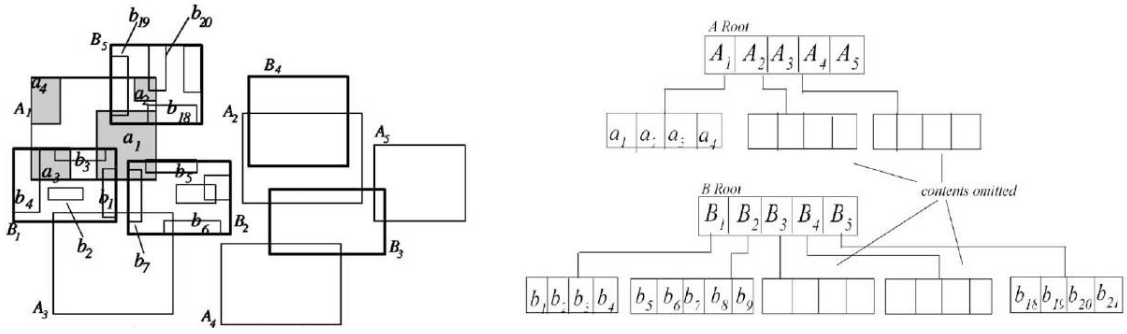


Figure 4.3: Example R-tree; Best Fit and Threshold computations for join operator.

4.5.4.2 Query Processing for Definable Join Conditions

Also for join the definable condition is approximated by a set of well-defined conditions. Thus, processing for join operations based on definable conditions corresponds to executing a set of join operations based on well-defined conditions and applying a given operation to the resulting sets, according to what presented in Section 4.4.2.2.

We refer to the algorithms for Best Fit and Thresholds semantics for well-defined join conditions are exploited for the execution based on definable conditions. Notice that we suppose to perform the calculation of the well-defined selection conditions as discussed and then to pass them to the algoritms.

θ	$\mathbf{d(O.r)}$	$\mathbf{d(ft), d(O)}$	$\mathbf{c}(\theta, \mathbf{d(ft)}, \mathbf{d(O)})$	$\mathbf{co}(\theta, \mathbf{d(ft)}, \mathbf{d(O)})$
d	2	$(*, 2)$	d	d
		$(*, 1)$	d	d
	0	$(*, 0)$	d	d
t	2	$(*, 2)$,$(2, 1)$	d, t	d, t
		$(0, 1)$	$\mathcal{T}_{0,1}$	$\mathcal{T}_{0,1}$
		$(1, 1)$	d, t, r, o	d, t, r, o
	0	$(2, 0)$	d, t	d, t
		$(1, 0)$	$\mathcal{T}_{1,0}$	$\mathcal{T}_{1,0}$
		$(0, 0)$	$\mathcal{T}_{0,0}$	
i	2	$(2, 2)$	d, t, i, b, o	d, t, i, b, o
		$(1, 2)$	d, t, i, b, r	d, t, i, b, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)
		$(2, 1)$	d, t, r	d, t, r
		$(1, 1)$	d, t, i, b, r, o	d, t, i, b, r, o
b	2	$(2, 2)$	d, t, i, b, o	d, t, b, o
		$(1, 2)$	d, t, i, b, r	d, t, i, b, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	
		$(2, 1)$	d, t, r	d, t, r
		$(1, 1)$	d, t, i, b, r, o	d, t, i, b, r, o
e	2	$(2, 2)$	d, t, i, b, e, v, o	t, b, e, v, o
		$(1, 2)$	d, t, i, b, r	t, i, b, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	
		$(2, 1)$	d, t, c, v, r	t, c, v, r
		$(1, 1)$	$\mathcal{T}_{1,1}$	t, i, b, e, c, v, r, o
c	2	$(2, 2)$	$\mathcal{T}_{2,2}$	d, t, c, v, o
		$(1, 2)$	d, t, i, b, r	d, t, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	
		$(2, 1)$	d, t, c, v, r	d, t, c, v, r
		$(1, 1)$	$\mathcal{T}_{1,1}$	d, t, c, r, o, v
	0	$(2, 0)$,$(1, 0)$	$\mathcal{T}_{2,0}$ ($\mathcal{T}_{1,0}$)	$\mathcal{T}_{2,0}$ ($\mathcal{T}_{1,0}$)
		$(0, 0)$	$\mathcal{T}_{0,0}$	
v	2	$(2, 2)$	d, t, i, b, e, v, o	d, t, v, o
		$(1, 2)$	d, t, i, b, r	d, t, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	
		$(2, 1)$	d, t, c, v, r	d, t, c, v, r
		$(1, 1)$	$\mathcal{T}_{1,1}$	d, t, c, v, r, o
o	2	$(2, 2)$	d, t, i, b, o	d, t, o
		$(1, 2)$	d, t, i, b, r	d, t, r
		$(0, 2)$,$(0, 1)$	$\mathcal{T}_{0,2}$ ($\mathcal{T}_{0,1}$)	
		$(2, 1)$	d, t, r	d, t, r
		$(1, 1)$	d, t, i, b, r, o	d, t, r, o

Table 4.4: Compatibility sets for selection operator. In column 3, the notation $(*, d)$ corresponds to: $(0, d)$, $(1, d)$ and $(2, d)$.

Nearest Neighbor semantics and Threshold are presented together since they share the same operation. In particular:

- the algorithms should take as input $\mathcal{F}_j(C)$;
- the computation of the topological similarity intervals (lines 15, 28, 40 and 52 in Algorithm 7, lines 16, 25, 33, 39 and 41 in Algorithm 8 and Algorithm 9) is substituted by the computation of n topological similarity intervals $tsi(e_a, \theta_i^q, e_b)$, $i = 1, .., n$, followed by an aggregate operations over such intervals.

Best Fit semantics. The maximum interval is computed:
$(max\{tsi(e_a, \theta_i^q, e_b).1 | i = 1, ..., n\}, max\{tsi(e_a, \theta_i^q, e_b).2 | i = 1, ..., n\})$ See Algorithm 7.

Threshold semantics and Nearest Neighbor. The union of intervals is computed:

$(min\{tsi(e_a, \theta_i^q, e_b).1 | i = 1, ..., n\}, max\{tsi(e_a, \theta_i^q, e_b).2 | i = 1, ..., n\})$ See Algorithm 8 and Algorithm 9.

4.6 Summary

In this chapter, we have presented a general approach for relaxing topological selection and join operators.

First of all, as already discussed, since not all topological predicates are defined for each pair of feature types, two distinct types of selection and join conditions have been considered for relaxation:

- *Well-defined*: they rely on topological relations which are defined for the objects used in the condition.
- *Definable*: definable conditions are non well-defined because they rely on a topological relation that is not defined for the spatial dimension of the considered feature types, but they would become well-defined by changing such dimensions. Thus, it seems reasonable to assign a semantics to the query anyway, by rewriting the query condition using the most similar topological relations defined for the dimension of features in the available data. For this reason, we say that the condition is *definable* since the syntax error can be "recovered" by query rewriting.
- *Incorrect*, if they are neither well-defined nor definable.

Then, we presented the overall relaxation strategy that is based on: (i) a relaxation predicate $rp()$, that applies a certain degree of relaxation in selecting result objects (or objects pairs for join); (ii) a similarity function among topological relations, which provides a relaxation measurement, we consider in particular the function introduced in Section 3.2.3. Different definitions for the relaxation predicate provide different semantics for the relaxed operators. In this chapter, we proposed three semantics for $rp()$: two (Best Fit and Threshold) for both well-defined and definable conditions and an additional one for definable conditions (Nearest Neighbor). The Best Fit semantics applies the minimum amount of relaxation to the query condition in order to return a non-empty answer. Thus, it models a variation of a top-1 query dealing with spatial relations. The Threshold semantics relaxes the topological query up to a certain fixed limit, depending on system parameters. While under the Threshold semantics relaxation is always applied and, if required, can be

hardcoded, in the Best Fit semantics relaxation depends on the input dataset and is applied only if no spatial objects satisfy the query condition (thus, guaranteeing a non empty relaxed result). The Nearest Neighbor semantics, defined only for definable conditions, relaxes the topological condition with its closest topological relations.

Query processing algorithms for the execution of relaxed topological selection and join operators under the Best Fit, the Threshold and Nearest Neighbor semantics, based on well-defined and definable conditions, have finally been provided. The proposed algorithms rely on the usage of the R-tree index structure and are based on a branch and bound approach, which discards the visit of some R-tree sub-trees that cannot produce further results. The pruning condition is defined by using a topological similarity function and some compatibility rules, first proposed in [PTSE95] for regions, here extended to pairs of spatial objects of arbitrary dimension. In Chapter 5, we will present a prototype system, called ARTjQA (Advanced Relaxed Topological java Query Analyzer), implementing most of the algorithms presented in this chapter, in order to evaluate their efficacy and efficiency.

θ	$(\mathbf{d}(\mathbf{e_a}.\mathbf{r}), \mathbf{d}(\mathbf{e_b}.\mathbf{r}))$	$(\mathbf{d}(\mathbf{ft_a}), \mathbf{d}(\mathbf{ft_b}))$	$\mathbf{c}(\theta, \mathbf{d}(\mathbf{ft_a}), \mathbf{d}(\mathbf{ft_b}))$	$\mathbf{co}(\theta, \mathbf{d}(\mathbf{ft_a}), \mathbf{d}(\mathbf{ft_b}))$
d	$(2,2)$	$(2,2)$, $(2,1)$, $(1,2)$, $(1,1)$	d	d
		$(0,2)$, $(0,1)$	d	
		$(*,0)$	d	
	$(2,0)$	$(2,0)$, $(1,0)$		d
	$(0,2)$	$(0,2)$, $(0,1)$		d
	$(0,0)$	$(0,0)$		d
t	$(2,2)$	$(2,2)$, $(1,2)$	d, t	d, t
		$(0,2)$, $(2,0)$	d, t	
		$(0,1)$	$\mathcal{T}_{0,1}$	
		$(2,1)$	d, t	d, t
		$(1,1)$	d, t, r, o	d, t, r, o
		$(1,0)$	$\mathcal{T}_{1,0}$	
		$(0,0)$	$\mathcal{T}_{0,0}$	
	$(2,0)$	$(2,0)$		d, t
		$(1,0)$		$\mathcal{T}_{1,0}$
	$(0,2)$	$(0,2)$		d, t
		$(0,1)$		$\mathcal{T}_{0,1}$
i	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	d, t, i, b, o
		$(1,2)$	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2}$
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	d, t, r
		$(1,1)$	$\mathcal{T}_{1,1}$	d, t, i, b, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	
	$(0,2)$	$(0,2)$, $(0,1)$		$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$
b	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	d, t, b, o
		$(1,2)$	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2}$
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	d, t, r
		$(1,1)$	$\mathcal{T}_{1,1}$	d, t, i, b, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	
e	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	t, b, e, v, o
		$(1,2)$	$\mathcal{T}_{1,2}$	t, b, r, i
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	t, v, r , c
		$(1,1)$	$\mathcal{T}_{1,1}$	t, i, b, e, c, v, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	e
	$(0,0)$	$(0,0)$		e
c	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	d, t, c, v, o
		$(1,2)$	$\mathcal{T}_{1,2}$	d, t, r
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	$\mathcal{T}_{2,1}$
		$(1,1)$	$\mathcal{T}_{1,1}$	d, t, c, v, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	
	$(2,0)$	$(2,0)$, $(1,0)$		$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$
v	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	d, t, v, o
		$(1,2)$	$\mathcal{T}_{1,2}$	d, t, r
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	$\mathcal{T}_{2,1}$
		$(1,1)$	$\mathcal{T}_{1,1}$	d, t, c, v, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	
o	$(2,2)$	$(2,2)$	$\mathcal{T}_{2,2}$	d, t, o
		$(1,2)$	$\mathcal{T}_{1,2}$	d, t, r
		$(0,2)$, $(0,1)$	$\mathcal{T}_{0,2}$ $(\mathcal{T}_{0,1})$	
		$(2,1)$	$\mathcal{T}_{2,1}$	d, t, r
		$(1,1)$	$\mathcal{T}_{1,1}$	d, t, r, o
		$(2,0)$, $(1,0)$	$\mathcal{T}_{2,0}$ $(\mathcal{T}_{1,0})$	
		$(0,0)$	$\mathcal{T}_{0,0}$	

Table 4.5: Compatibility sets for join operator. In column 3, the notation $(*, d)$ corresponds to: $(0, d)$, $(1, d)$ and $(2, d)$.

Algorithm 7 Algorithms for Join: BF semantics.

INPUT
Feature types ft_a, ft_b
Query relation θ^q
R-tree R_a over ft_a instances in map M_a
R-tree R_b over ft_b instances in map M_b
Ensure: Objects in $\bowtie_{ft_a \theta_q ft_b}$ under the Best Fit semantics
METHOD
PriorityQueue H:= empty
H.insert($R_a.root$,$R_b.root$,0,0)
pc:=0
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem.e_a$, $elem.e_b$ are both intermediate entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q, e'_b)
        if $simmin > pc$ **then**
          $pc := simMin$
          clearQueue(H,pc)
        end if
        if $simmax >= pc$ **then**
          H.insert(e'_a,e'_b,$simmin$,$simmax$)
        end if
      end for
    end for
  else if $elem.e_a$ is an intermediate entry,$elem.e_b$ is a leaf entry **then**
    R_a is higher than R_b
    for all $e'_a \in elem.e_a.n$ **do**
      ($simmin$,$simmax$):=tsi(e'_a,θ^q,$elem.e_b$)
      if $simmin > pc$ **then**
        $pc := simMin$
        clearQueue(H,pc)
      end if
      if $simmax >= pc$ **then**
        H.insert(e'_a,$elem.e_b$,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_b$ is an intermediate entry,$elem.e_a$ is a leaf entry **then**
    R_b is higher than R_a
    for all $e'_b \in elem.e_b.n$ **do**
      ($simmin$,$simmax$):=tsi($elem.e_a$,θ^q,e'_b)
      if $simmin > pc$ **then**
        $pc := simMin$
        clearQueue(H,pc)
      end if
      if $simmax >= pc$ **then**
        H.insert($elem.e_a$,e'_b,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_a$,$elem.e_b$ are leaf entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q,e'_b)
        if $simmax >= pc$ **then**
          determine θ^m between $e'_a.o$ and $e'_b.o$
          $RankVal$:=ts($\theta^m_{d(ft_a),d(ft_b)}$, $\theta^q_{d(ft_a),d(ft_b)}$)
          if $RankVal = pc$ **then**
            H.insert($e'_a.o$,$e'_b.o$,$RankVal$,$RankVal$)
          end if
          if $RankVal > pc$ **then**
            $pc := RankVal$
            clearQueue(H,pc)
            H.insert($e'_a.o$,$e'_b.o$,$RankVal$,$RankVal$)
          end if
        end if
      end for
    end for
  else
    output $elem.e_a$, $RankVal$
  end if
end while

Algorithm 8 Algorithms for Join: Thr semantics.

INPUT
Feature types ft_a, ft_b
Query relation θ^q
R-tree R_a over ft_a instances in map M_a
R-tree R_b over ft_b instances in map M_b
Threshold ρ
Ensure: Objects in $\bowtie_{ft_a \theta_q ft_b}$ under the Threshold semantics
METHOD
PriorityQueue H:= empty
H.insert($R_a.root$,$R_b.root$,0,0)
pc:=ρ
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem.e_a$, $elem.e_b$ are intermediate entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q,e'_b)
        if $simmax >= pc$ **then**
          H.insert(e'_a,e'_b,$simmin$,$simmax$)
        end if
      end for
    end for
  else if $elem.e_a$ is an intermediate entry,$elem.e_b$ is a leaf entry **then**
    R_a is higher than R_b
    for all $e'_a \in elem.e_a.n$ **do**
      ($simmin$,$simmax$):=tsi(e'_a,θ^q,$elem.e_b$)
      if $simmax >= pc$ **then**
        H.insert(e'_a,$elem.e_b$,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_b$ is an intermediate entry,$elem.e_a$ is a leaf entry **then**
    R_b is higher than R_a
    for all $e'_b \in elem.e_b.n$ **do**
      ($simmin$,$simmax$):=tsi($elem.e_a$,θ^q,e'_b)
      if $simmax >= pc$ **then**
        H.insert($elem.e_a$,e'_b,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_a$, $elem.e_b$ are leaf entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q,e'_b)
        if $simmax >= pc$ **then**
          determine θ^m between e'_a and e'_b
          $RankVal$:=ts($\theta^m_{d(ft_a),d(ft_b)}$, $\theta^q_{d(ft_a),d(ft_b)}$)
          if $RankVal >= pc$ **then**
            H.insert($e'_a.o$,$RankVal$,$RankVal$)
          end if
        end if
      end for
    end for
  else
    output $elem.e_a$, $RankVal$
  end if
end while

Algorithm 9 Algorithms for Join: NN semantics.

INPUT
Feature types ft_a, ft_b
Query relation θ^q
R-tree R_a over ft_a instances in map M_a
R-tree R_b over ft_b instances in map M_b
Threshold ρ
Ensure: Objects in $\bowtie_{ft_a \theta_q ft_b}$ under the Threshold semantics
METHOD
PriorityQueue H:= empty
H.insert($R_a.root$,$R_b.root$,0,0)
pc:=1
while H is not empty **do**
  $elem$:=H.getMax()
  if $elem.e_a$, $elem.e_b$ are intermediate entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q,e'_b)
        if $simmax = pc$ **then**
          H.insert(e'_a,e'_b,$simmin$,$simmax$)
        end if
      end for
    end for
  else if $elem.e_a$ is an intermediate entry,$elem.e_b$ is a leaf entry **then**
    R_a is higher than R_b
    for all $e'_a \in elem.e_a.n$ **do**
      ($simmin$,$simmax$):=tsi(e'_a,θ^q,$elem.e_b$)
      if $simmax = pc$ **then**
        H.insert(e'_a,$elem.e_b$,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_b$ is an intermediate entry,$elem.e_a$ is a leaf entry **then**
    R_b is higher than R_a
    for all $e'_b \in elem.e_b.n$ **do**
      ($simmin$,$simmax$):=tsi($elem.e_a$,θ^q,e'_b)
      if $simmax = pc$ **then**
        H.insert($elem.e_a$,e'_b,$simmin$,$simmax$)
      end if
    end for
  else if $elem.e_a$, $elem.e_b$ are leaf entries **then**
    for all $e'_a \in elem.e_a.n$ **do**
      for all $e'_b \in elem.e_b.n$ **do**
        ($simmin$,$simmax$):=tsi(e'_a,θ^q,e'_b)
        if $simmax = pc$ **then**
          determine θ^m between e'_a and e'_b
          $RankVal$:=ts($\theta^m_{d(ft_a),d(ft_b)}$, $\theta^q_{d(ft_a),d(ft_b)}$)
          if $RankVal = pc$ **then**
            H.insert($e'_a.o$,$RankVal$,$RankVal$)
          end if
        end if
      end for
    end for
  else
    output $elem.e_a$, $RankVal$
  end if
end while

Chapter 5

Experimental Results

In order to evaluate the efficacy and efficiency of the query processing techniques presented in Chapter 4, we developed a prototype system, called ARTjQA (Advanced Relaxed Topological java Query Analyzer). The aim of this chapter is to present the architecture of such prototype, the characteristics of the datasets we used for the experimentation activity, and the results we have obtained. More precisely, the modules of the prototype and their interactions are presented in Section 5.1. The characteristics of the synthetic datasets as well as details and considerations about the set up of the experiments are presented in Section 5.2. The obtained results are finally discussed in Sections 5.3 and 5.4 for both relaxed selection and join operators, under all the proposed semantics.

5.1 The ARTjQA Prototype

ARTjQA (Advanced Relaxed Topological java Query Analyzer) is a prototype system developed in Java 6, implementing all the query processing techniques presented in Chapter 4. In the following, we briefly review the ARTjQA architecture, pointing out the main libraries ARTjQA relies on (Section 5.1.1) and its main core components (Section 5.1.2).

5.1.1 The External Modules

ARTjQA relies on two main existing libraries: (i) one for implementing the used index structure; (ii) one for implementing topological relationships.

Concerning the index structure, we use an R*-tree since, based on the experiments reported in [PTSE95], it provides the best average performance independently of the size of objects

in the dataset when executing topological operations based on an arbitrary relation. In particular, we used the R*-tree implementation available at http://research.att.com/marioh/spatialindex/, referred in the following as *spatial index library*.

The spatial index library consists of the following packages:

- `storagemanager`, that provides a common interface for index storage management;
- `spatialindex`, that provides the basic index structures and the methods to efficiently access geo-spatial information;
- the `rtree`, that implements the R-tree index and its variants. The default index is the R*-tree.

We have slightly modified the implementation of the package `rtree` by changing the visibility of some classes in order to make them accessible from our prototype. Since the index is balanced nodes can be under full. They cannot be empty though. A fill factor specifies the minimum number of entries allowed in each node. The fill factor is usually close to 70%.

For the implementation of topological predicates, we used JTS (Java Topology Suite[1]), a Java API which conforms to the Simple Features Specification for SQL (SFS) published by the Open GIS Consortium ([OGCa]) and provides a complete, consistent, and robust implementation of a fundamental set of 2D geo-spatial data operations. As claimed in the documentation, JTS attempts to implement the OpenGIS SFS as accurately as possible. When SFS is unclear or omits some details, JTS chooses a reasonable and consistent alternative. We used JTS for two main purposes:

- For geo-spatial data encoding, inside the R*-tree implementation. In particular, JTS objects have been stored inside the index using a conversion from Java objects to bytes arrays.
- For implementing topological predicates. Topological predicates in JTS are implemented as binary predicates, according to OGC Simple Features Specification. We have slightly modified the API in order to: (i) return false, instead of undefined, for combinations of input geo-spatial data which are not in the domain of a predicate (e.g. touches(Point, Point) => false); (ii) make topological predicates mutually exclusive.

  We have further modified JTS in order to implement the set of topological relations presented in Section 3.2.1: $\mathcal{T}$={ `disjoint (d), touch (t), in (i), contains`

[1] http://www.vividsolutions.com/jts/jtshome.htm

	Original pattern defined in JTS	New pattern defined in JTS	
Name	**Pattern of the 9-int. matrix**	**Object dimension**	**Pattern of the 9-int. matrix**
`disjoint (d)`	$FF*-FF*-***$	d_1/d_2, with $d_1 \neq 0$, $d_2 \neq 0$,	$FFT-FFT-T*T$
		d_1/d_2, with $d_1 = 0$ or $d_2 = 0$,	$FFT-FFF-T*T$
`touch (t)`	$FT******$	2/2, 2/1, 1/2, 1/1	$F**-*T*-**T$
		2/1, 2/0, 1/1, 1/0 1/2, 1/1, 0/1, 0/2	$F**-TF*-**T$ $FT*-FF*-**T$
`in (i)`	$T*F-**F-***$	2/2, 1/1, 1/2, 0/2, 0/1	$T*F-*FF-TTT$
contain (c)	$T**-***-FF*$	2/2, 2/1, 2/0, 1/1, 1/0	$T*T-*FT-FFT$
`equal (e)`	$T*F-**F-FF*$	2/2, 1/1,	$TFF-FTF-FFT$
		0/0	$TFF-FFF-FFT$
`cross (r)`	$T*T-***-***$	1/2, 2/1	$T*T-***-T*T$
		1/1	$0*T-***-T*T$
`overlap (o)`	$T*T-***-T**$	2/2	$T*T-***-T*T$
		1/1	$1*T-***-T*T$
`cover (v)`	–	2/2, 2/1, 1/1	$T*T-*TT-FFT$
`coveredby (b)`	–	2/2, 1/1, 1/2	$T*F-*TF-TTT$

Table 5.1: Definition of the reference set of topological relations.

`(c), coveredby (b), covers (v), equal (e), overlap (o)` }. Table 5.1, for each topological predicate, presents the semantics originally implemented by JTS and the one we have implemented.

5.1.2 The Core ARTjQA Modules

The ARTjQA prototype is a configurable application for executing topological relaxed selection and join queries against geo-spatial datasets.

ARTjQA relies on a modular architecture, which makes it easily extensible and modifiable. Each module is independent and communicates with the other modules through message exchanges, represented as specific Java objects. Each module takes in input a particular object, performs a specific operation on the data contained in the object, creates and returns in output another object representing the result of the computation.

Since the approaches we have proposed for relaxed selection and join operators are independent on the considered compatibility rules and similarity function (see Sections 3.2.3 and 4.5), ARTjQA has been designed in a configurable way with respect to such information. In particular, such information can specified through an input XML file. Both compatibility rules and similarity values should be provided using a tabular format. An example of configuration file is proposed in Figure A.1 in Appendix A.

Similarly to the configuration file, also the input to the application, corresponding to the

set of queries to be executed, can be specified using an XML format. The file has a general container tag `<job>`. Each query is then specified inside tag `<query>`, providing the following information:

- type of query (selection/join);
- information on the R*-tree to be used (path and name);
- information on the indexed dataset (path and name);
- the type of search to be considered (with or without the index);
- the type of semantics to be used in the processing (Best Fit, Threshold, Nearest Neighbor, Non Relaxed), and eventually the threshold value, only for the Threshold semantics;
- the topological predicates (in order to improve the flexibility of ARTjQA prototype, we allow to specify more than one predicate separated by comma, this possibility is allowed only for definable conditions since the set of predicates represent the well defined condition to execute) of the query condition;
- the path about the result files where collected query processing statistics have to be written.

An example of the input file is presented in Figures A.2 and A.3 in Appendix A, for both selection and join operators.

The prototype is composed of five modules (see Figure 5.1):

- *Configuration module*: the set of classes that take in input an XML configuration file and use its content to configure compatibility and similarity tables.
- *Validator module*: the set of classes that control correctness of the content of the XML input file.
- *Data loader*: the set of classes that construct, load, and return the data structures representing the queries to be executed.
- *Processing module*: the set of classes executing the set of queries described in the input file and computing several statistics.
- *Result maker module*: the set of classes that, using the statistics collected during query processing, create a set of result files and store them in the path given in input.

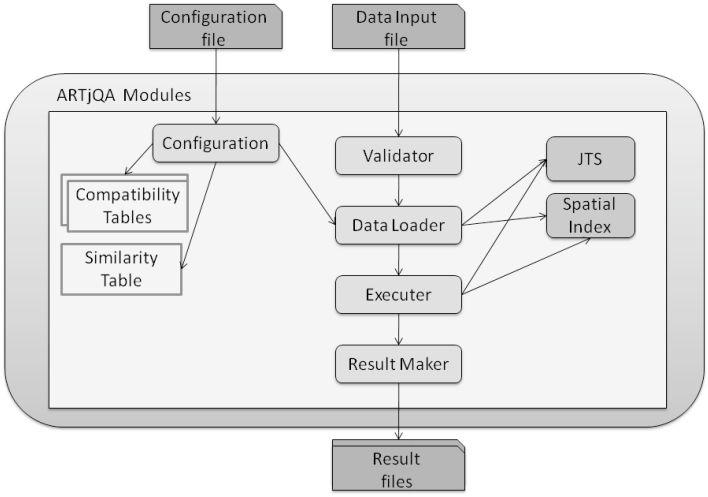


Figure 5.1: ARTjQA prototype.

5.2 Experimental Setup

Experiments were performed on a Windows machine, with processor Intel Core Duo CPU T9300 2,5GHz and 4Gb of main memory. Similarly to other performed experimentation ([ZPZL05, JS07]), the R*-tree has been configured in order to get a random buffer that accommodates about the 10% of the tree, and a page size of 4KBytes.

In performing experiments, we considered an istance of the ARTjQA prototype based on the compatibility tables and the similarity function defined in Sections 3.2.3 and 4.5, respectively. Starting from the consideration that it is not reasonable for a real application to return as result to a user query all the objects that are *disjoint* from a query or another dataset object (unless, this is explicitly requested by the user), compatibility tables have been modified by removing *disjoint* from the set of compatible relations of all topological relations except *disjoint* (since in this case *disjoint* is explicitly requested) and *touch* (since, based on the used similarity function, in this case *disjoint* is the most similar relation).

Experiments have been performed on randomly generated datasets. This choice ensures a very high flexibility which cannot be guaranteed by using the real geo-spatial datasets at our disposal. Using adequate synthetic datasets, we created various scenarios, stressing different relevant aspects of relaxed query processing, such as the dimension, the number, and the distribution of geo-spatial data.

Synthetic datasets have been generated using the useful tool available at this web link

http://www.rtreeportal.org/software/SpatialDataGenerator.zip. Such tool allows one to create synthetic datasets of (multi-dimensional) rectangles, contained in a reference space of configurable size, between 1 and 100000. The size of the space defines the upper bounds of the random x, y, z coordinates for each box. In the performed experiments, we considered 2-dimensional rectangles inside a 1000×1000 reference space.

We then considered three distinct object dimensions: small MBRs, medium MBRs and large MBRs, whose size is at most 0.5 %, 1 % and 3 % of the global area, respectively. For each object size, we constructed 10 datasets (containing respectively $10.000K$, $K = 1, ..., 10$, MBR's), based on different distributions. Three different distributions have been considered: (i) uniform distribution; (ii) Gaussian distribution, with $mean = 500$ and $sigma = 100$; (iii) skewed distribution, with $p = 1$

We implemented the following techniques:

- IBF_θ: index-based processing for BF semantics with relation θ, for both selection and join;
- ITHR(ρ)_θ: index-based processing for Thr semantics with ρ as threshold and relation θ, for both selection and join;
- INN_θ, index-based processing for Nearest Neighbor semantics with relation θ, for both selection and join;
- SBF_θ: sequential processing for selection under BF semantics with relation θ;
- STHR(ρ)_θ: sequential processing for selection under Thr semantics with ρ as threshold and relation θ;
- INR_θ: non relaxed index-based processing for selection with relation θ, for both selection and join.

Collected statistics are the following:

- *Response time* (denoted by *Time (msec)* in result graphs): it represents the response time of executing a query, without considering the time required for the construction of the R*-trees, the parsing of the input file, and the construction of the data structures required for query execution. It is computed for all implemented query processing techniques, for both selection and join.
- *(Pairs of) Visited nodes* (denoted by *(% (Pairs of) Visited nodes) x 100*): it is the rate between the number of visited nodes (node pairs) with respect to the total number of nodes (node pairs). It is computed only for index-based query processing techniques, for both selection and join.

- *Pairs of Visited features* (denoted by *(% (Pairs of) Visited features) x 100*): it corresponds to the number of visited leaf MBRs in the R*-tree with respect to the total number of features pairs. It is computed only for index-based query processing techniques, only for join.

- *(Pairs of) Compatible features* (denoted by *(% (Pairs of) Compatible features) x 100* in result graphs): it is the rate between the number of visited leaf MBRs (MBR pairs) with respect to the total number of features (features pairs). Again, only for index-based techniques.

In order to better understand the proposed results, it is important to notice that the frequency of the topological relations we consider can vary depending on several situations. Generally speaking, it is possible to divide the topological predicates into: *high selective* predicates and *low selective* predicates. In particular, high selective predicates are those based on the following topological relations: *cover*, *coveredby*, *in*, *contain*, *equal*, *touch*; such relations are not very frequent among objects in real geo-spatial maps. On the other side, low selective predicates are those based on the *disjoint* and *overlap* relations; indeed such relations are very common among objects in real geo-spatial maps. Some possible increase/decrease of the frequency of predicates depends on the dimensions of the considered objects. In particular, if the first (the second) object involved in the topological relation is smaller than the second (the first), relations *in* and *cover* (*contain* and *cover*) become more frequent.

Another importance issue in designing experiments concerns the criteria for choosing the threshold values in order to verify their impact on the computation of operators based on the Thr semantics. For all relations except *disjoint* and those do not satisfying property **P2**, as defined in Section 3.2.3 (*touch* for the considered similarity function), the chosen values guarantee the following behavior during the relaxed evaluation: (i) v_{no_dj} it is the maximum similarity value for which *disjoint* is always discarded in the relaxed evaluation (in some cases it also excludes other relations), thus, it guarantees a high degree of relaxation; (ii) v_{P2} discards always *disjoint* and at least all the relations do not satisfying property **P2** (*disjoint* and *touch* in our case); (iii) v_3 keeps at most three topological relations during the relaxed evaluation; (iv) v_{no_relax} just keeps the query relation, thus it coincides with a similarity threshold equal to 1. Table 5.2 presents the chosen threshold values, used in the experimental evaluation, based on the similarity function presented in Section 3.2.3, applied to pairs of objects with various dimensions (pointed out in columns two, three, and four of the table).

In the experimental results reported in the following we always consider, if not differently specified, datasets and query objects whose size is at most the 3% of the global area (at most 30 units), We note that, among the considered dataset sizes, objects which size is at most 30 units lead to high computational costs, due to the higher object density in the

	Threshold value		
	(2,2)	**(1,1)**	**(2,1) (1,2)**
v_{no_dj}	0.5	0.75	0.61
v_{P2}	0.6	0.81	0.81
v_3	0.8	0.93	0.9
v_{no_relax}	1	1	1

Table 5.2: Threshold values.

reference space with respect to other data size. Thus, reported results can be interpreted as a worst case with respect to the other obtained results. The generated datasets are interpreted as MBRs of region objects. Similar results have been obtained by consider other object types and therefore additional topological relationships (i.e., *cross*, that is a low selective relation). We executed 10 times each relaxed query, considering as final result the average value of the statistics resulting from each of the 10 executions.

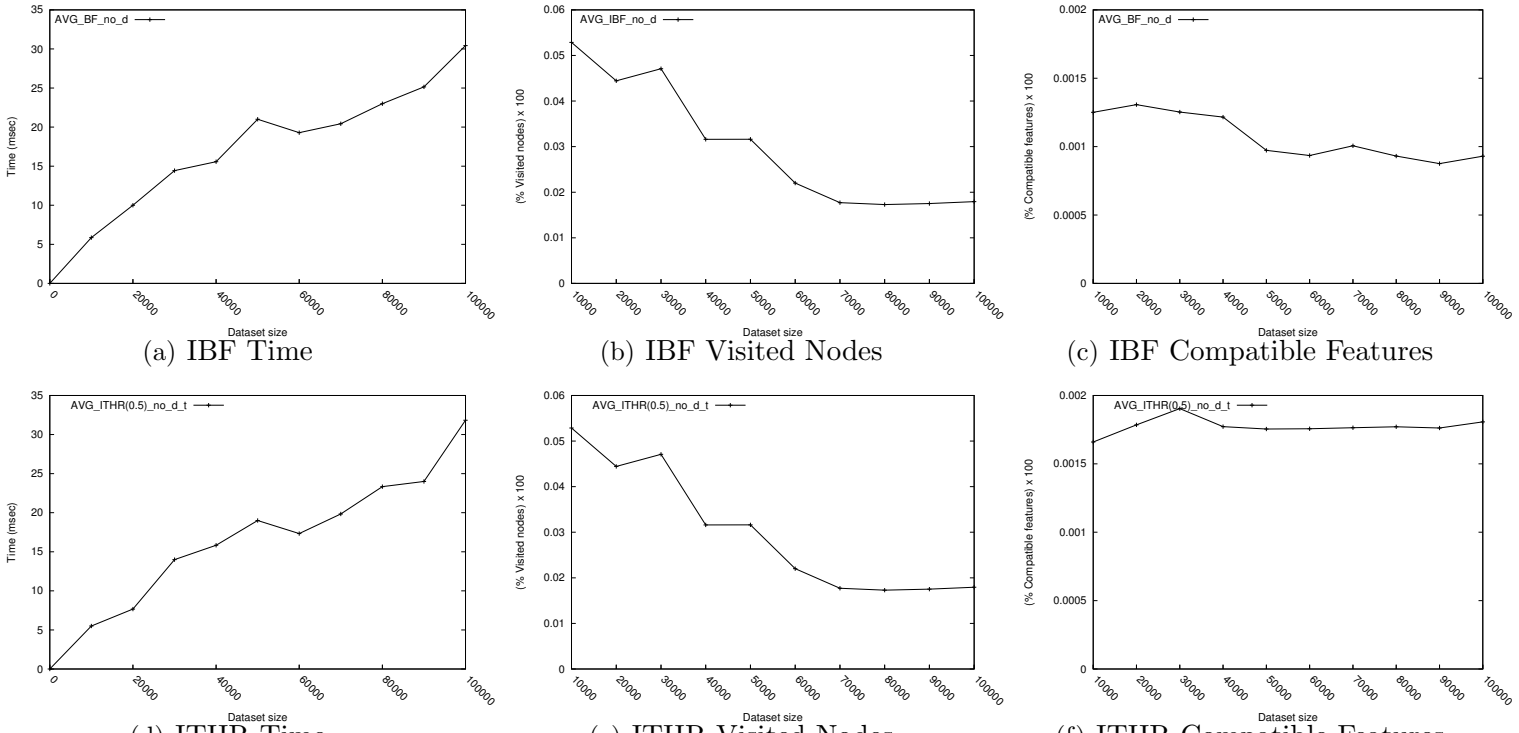
5.3 Results for Relaxed Selection Operator

In the following, we discuss the experimental results we have obtained by executing the relaxed selection operator against datasets with different distributions: uniform (Section 5.3.1), Gaussian and skewed (Section 5.3.2). The query objects we consider for selection operator are obtained from a real spatial object, the province of *imperia* in Liguria, a region of Italy, from which, we have obtained a region with 100 vertices. Starting from such region we have obtained objects with various dimensions, moreover for each dimensions we have computed ten objects positioned into ten different points of the reference space: two in the high left corner, two in the low left corner, two the center, two in the high right corner and two in the low right corner. Then for each position of the space we perform one query.

For each distribution, we have executed five groups of experiments. The aim of the first group is to compare the performance of indexed relaxed selection, denoted with IBF and ITHR, with respect to a sequential scan (only for uniform distribution). The aim of the second group of experiments is to compare the performance of indexed relaxed selection with respect to the execution of non relaxed selection, in order to analyze the impact of query relaxation. The third group of experiments aims at analyzing the impact of the query object size on query selectivity. The fourth group of experiments aims at analyzing the impact of the threshold value on performance. The fifth group of experiments analyzes performances in executing definable conditions.

5.3.1 Results for Uniform Distribution

In this section we report the results obtained for datasets with uniform distribution.



(a) IBF Time (b) IBF Visited Nodes (c) IBF Compatible Features

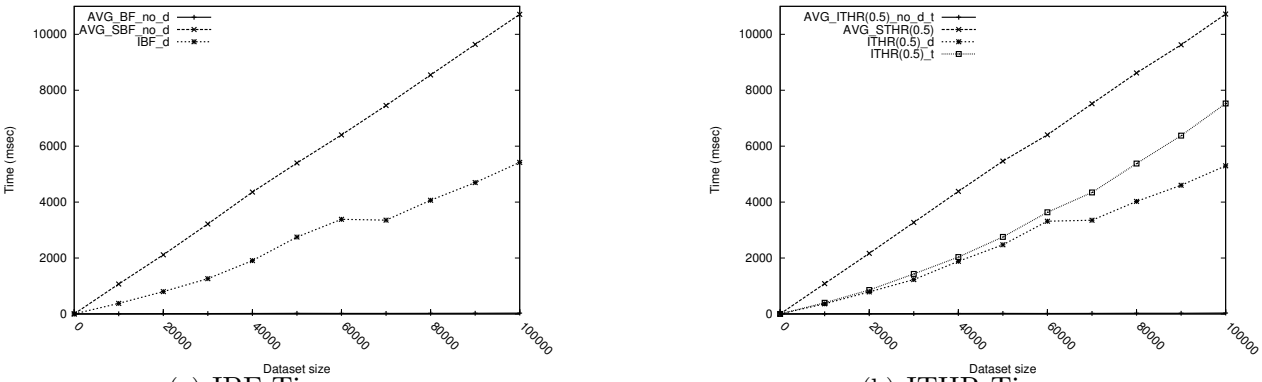
(d) ITHR Time (e) ITHR Visited Nodes (f) ITHR Compatible Features

Figure 5.2: Average performance for indexed relaxed selection with respect to sequential scan.

5.3.1.1 Indexed Relaxed Selection with respect to Sequential Scan

The aim of this group of experiments is to compare IBF_θ and SBF_θ as well as ITHR(ρ)_θ and STH(ρ)_θ in order to analyze the efficiency of the index based techniques with respect to a pure sequential scan as well as the cost in IBF and ITHR due to the visit of a larger portion of R*-tree. We considered $\rho = v_{no\_dj}$ (for the considered similarity function and object dimension, $v_{no\_dj} = 0.5$ - see Table 5.2), i.e., we consider the maximum relaxation value that guarantees to exclude only the *disjoint* relation from processing; using such value, under ITHR, the topological relation is relaxed with the highest number of relations.

Figure 5.2 (a) reports the average response time for IBF_θ, considering all relations but *disjoint*, denoted by AVG_IBF_no_d. Figure 5.2 (b) and (c) reports the average percentage of visited nodes and compatible features for AVG_IBF_no_d, respectively. Fig-



(a) IBF Time (b) ITHR Time

Figure 5.3: Indexed relaxed selection with respect to sequential scan.

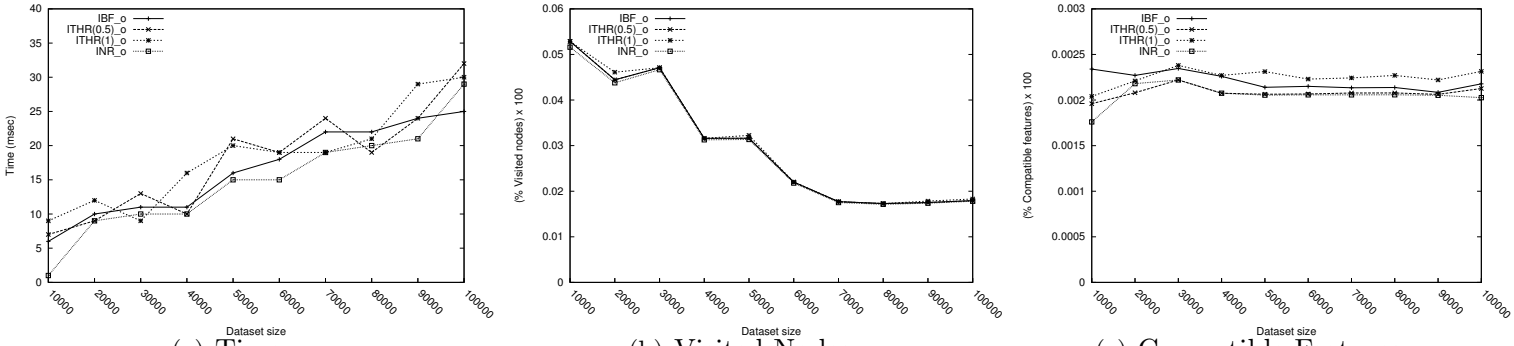
ure 5.2 (d) reports similar costs for ITHR(0.5)_θ, excluding *disjoint* and *touch* (we exclude also *touch* since its relaxations with $v_{no\_dj} = 0.5$ includes *disjoint*), denoted by AVG_ITHR(0.5)_no_d_t.

As expected, IBF_θ and ITHR(0.5)_θ show good performance in average, since also with datasets of high cardinality (100000), response time is under 35 msec. The trend for visited nodes and compatible features is decreasing, since we plot the average rate with respect to an increasing value (the cardinality of the dataset). Thus, since it is reasonable that both the average number of accessed features and visited nodes do not substantially change (since the query object size is the same), the average rate decreases with the increasing of dataset cardinality (and thus increasing the total number of node and features).

Figure 5.3(a) reports the comparison of average response times among: AVG_IBF_no_d, the sequential version of IBF_no_d, denoted by AVG_SBF_no_d; and the IBF_d. As expected, AVG_IBF_no_d and IBF_d are more efficient than AVG_SBF_no_d and SBF_d; indeed the curve of AVG_IBF_no_d is flattened on the horizontal axis and AVG_IBF_d has comparable but better response time w.r.t AVG_SBF_no_d.

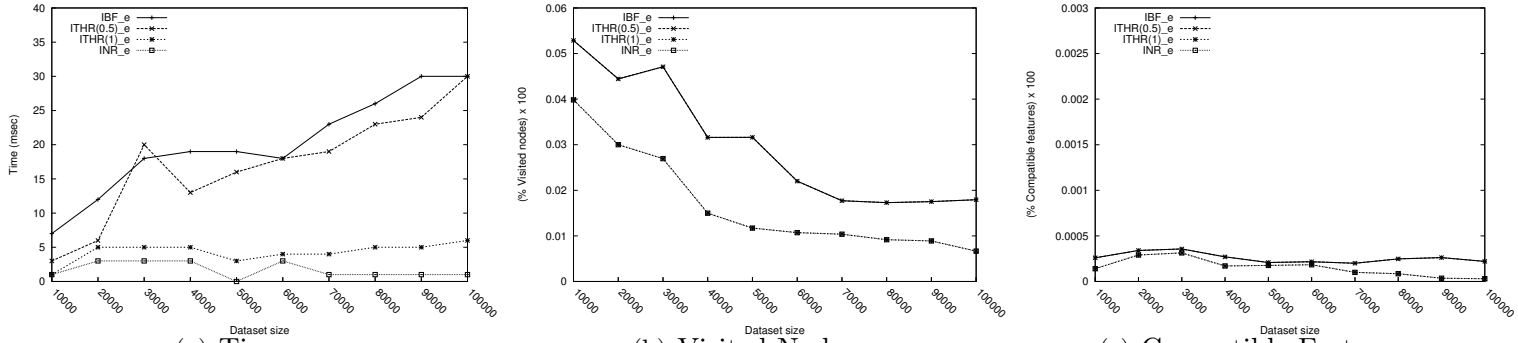
Figure 5.3(b) reports the average costs for ITHR(0.5)_θ excluding *disjoint* and *touch* (AVG_ITHR(v_{no_dj})_no_d_t) and the sequential version of ITHR(v_{no_dj})_θ, denoted by AVG_STHR(0.5), in addition we report also the response time for ITHR(0.5)_t and ITHR(0.5)_d in order to consider the worst case for ITHR(0.5)_θ. As expected, also AVG_ITHR(0.5)_no_d_t is more efficient than AVG_STHR(0.5); however, based on the chosen threshold value and the used similarity function, in this case *touch* behaves as *disjoint* (since, based on the used similarity function, *touch* relaxation includes *disjoint* and vice versa).

Since *disjoint* is the most frequent topological relation, even if it is not very significant in real applications, we decide not to consider it in the next experiments.



(a) Time (b) Visited Nodes (c) Compatible Features

Figure 5.4: Relaxed methods with respect to non relaxed ones for *overlap* (low selective relation).



(a) Time (b) Visited Nodes (c) Compatible Features

Figure 5.5: Relaxed methods with respect to non relaxed ones for *equal* (high selective relation).

5.3.1.2 Relaxed Methods with respect to Non Relaxed Ones

The aim of this group of experiments is to compare IBF_θ, ITHR_θ, and INR_θ. As threshold values, we consider v_{no_relax} (1 in our case) and v_{no_dj} (0.5 in our case), which, for the considered similarity function, is the value which excludes *disjoint* from the set of similar relations for all topological relations excluding *disjoint* and *touch*. We notice that since ITHR(1)_θ does not apply any relaxation, ITHR(1)_θ and INR always return similar results.

Value v_{no_dj} corresponds to the maximum relaxation and allows us to investigate how a deep relaxation impacts on the performance of the presented techniques. On the other side we consider v_{no_relax} in order to evaluate the additional costs of a relaxed threshold execution w.r.t. a non relaxed execution.

Figures 5.4 and 5.5 report results concerning selection conditions respectively based on a low selective relation, such as *overlap*, and an high selective relation such as *equal*. From

figures 5.4 and 5.5 we can see that relaxed techniques are less efficient than non relaxed ones for high selective relations (*equal* in our case), while for low selective predicates, such as *overlap*, differences in costs between relaxed and non relaxed processing techniques are less evident since the percentage of visited features increases also for INR_θ, thus increasing the overall costs due to the high costs of computing topological relations between selected features and the query object.

Figure 5.5 (a) shows that for high selective relations INR_θ and ITHR(1)_θ perform better than IBF_θ and ITHR(0.5)_θ since they correspond to more selective evaluations. The percentage of visited nodes (see Figure 5.5 (b)), which is higher for relaxed processing increases by increasing the number of topological relations taken into account (thus, ITHR(0.5)_θ is higher than ITHR(1)_θ); the percentage of visited features, as obvious, (see Figure 5.5(c)) increases by increasing the result set size.

On the other hand, for low selective predicates (*overlap*), as shown in Figures 5.4, response time, percetage of visited nodes and compatible features are similar for all the considered techniques. Indeed, in this case:

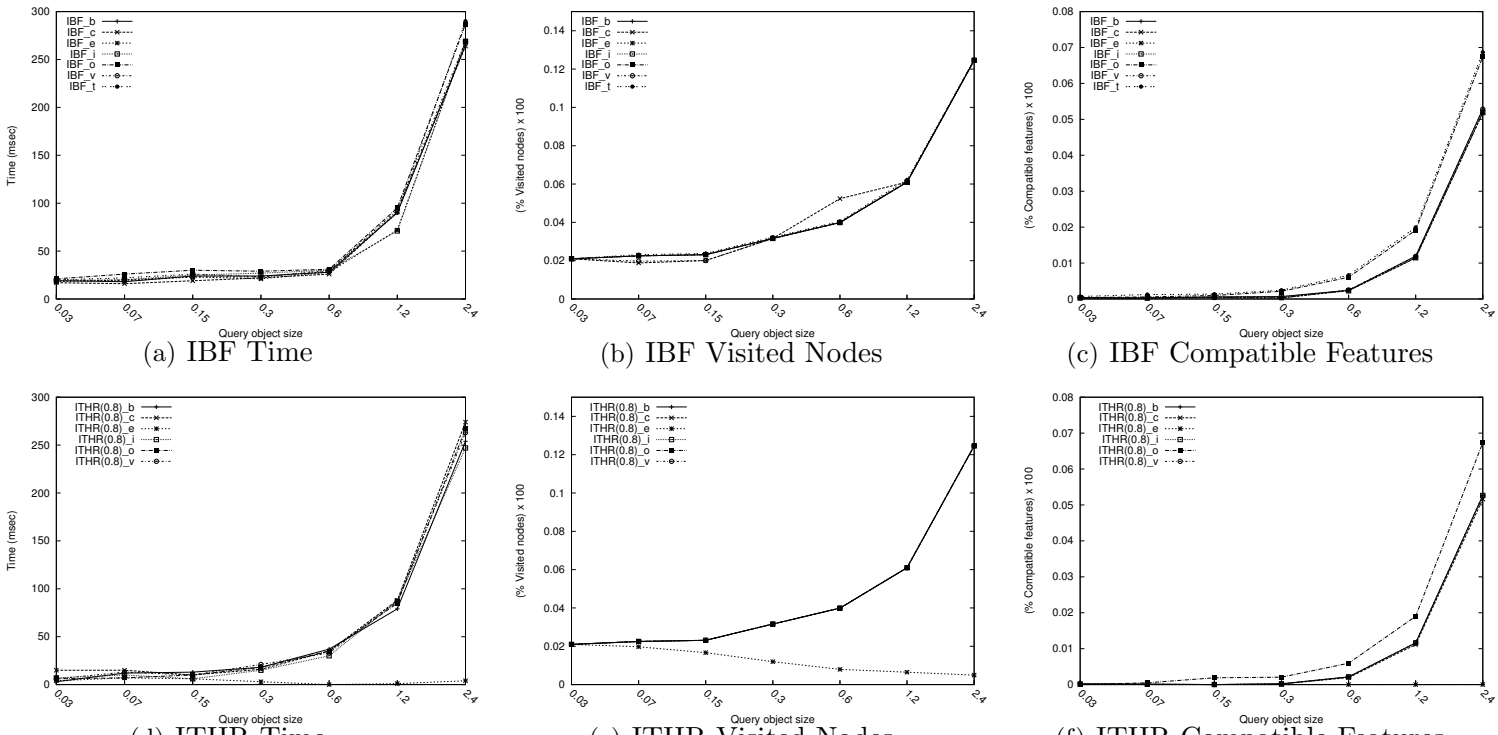
- it is highly probable that objects satisfying the query predicate exist, thus IBF$_o$ behaves like INR$_o$;
- since usually all predicates but *disjoint* are more selective than *overlap*, ITHR(0.5)_θ has a similar trend to INR_θ, since new relations considered by the relaxation do not significantly alter the result set.

5.3.1.3 Impact of the Query Object Size

For this group of experiments, we considered IBF_θ and ITHR(v_3)_θ (ITHR(0.8)_θ in our case) and we analyze the impact of object size on query selectivity for all relations but *disjoint* for IBF and all relations but *disjoint* and *touch* for ITHR(0.8)_θ .[2] We consider seven different query object sizes: 3, 7, 15, 30, 60, 120, 240 corresponding respectively to the 0.03 %, 0.07 %, 0.15 %, 0.3 %, 0.6 %, 1.2 % and 2.4 % of the global area.

As we can see from Figures 5.6, time increases while increasing the object size (thus, reducing selectivity). Differences in times for ITHR(0.8)_θ among the considered topological relations depends on the number of relations taken into account during relaxation based on the chosen threshold value. In particular, the *equal* relation has a very small execution time and the lowest percentage of accessed nodes and features since, for ITHR(0.8)_θ), *equal* relaxes into *equal* itself. Additionally, since it is a high selective relation, only very few objects satisfy it; this trend becomes more incisive while increasing the query object size

[2] we did not consider *touch* since it relaxes into disjoint.



(a) IBF Time (b) IBF Visited Nodes (c) IBF Compatible Features

(d) ITHR Time (e) ITHR Visited Nodes (f) ITHR Compatible Features

Figure 5.6: Impact of the query object size.

with respect to the dataset object dimensions. A similar behaviour holds when considering visited nodes and compatibility features. When considering visited nodes and compatible features for IBF, the trend is similar for all the considered relations.

5.3.1.4 Impact of Threshold Value

In order to analyze the impact of the threshold value, we report response time in msec for ITHR(ρ)_θ by considering four threshold values ($v_{no\_relax}, v_{P2}, v_3, v_{no\_dj}$). Table 5.3 presents the results obtained for relation *in*. Similar results have been obtained for other relations. Based on the used similarity function, for relation *in*, 7 relations are considered for value v_{no_dj}), 6 for value v_{P2}, 3 for value v_3, and of course 1 for value v_{no_relax}.

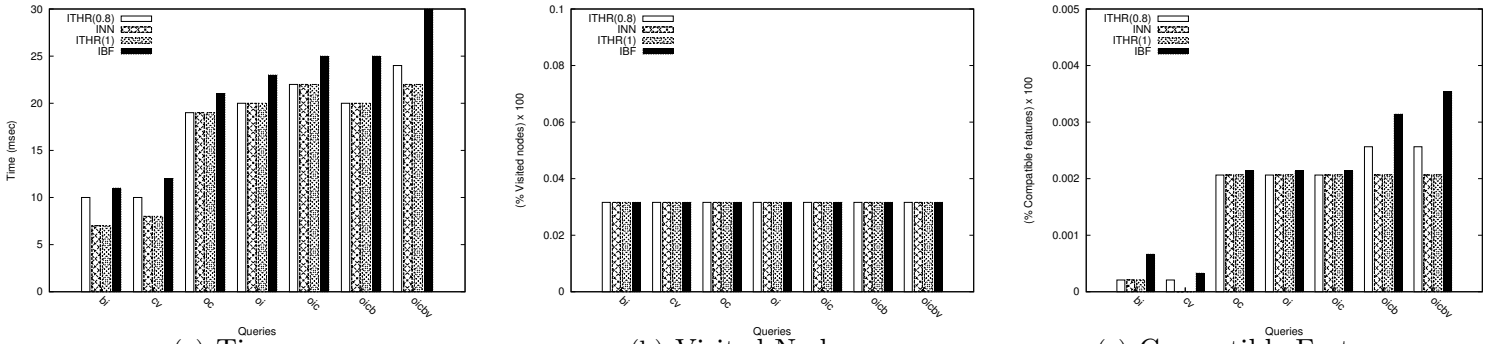
As expected, by increasing the threshold value performance increases since less relations are taken into account during the search. Specific results depend on the number of relations taken into account during processing, based on the chosen threshold value, and on the number of features that satisfy the selected relations.

There is no relevant difference in performance between the different values of the threshold, since the only very frequent relations is *disjoint*, while the other relations have almost the

same frequency.

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
10000	17	13	8	7	60000	31	29	25	23
20000	20	16	12	10	70000	33	30	25	23
30000	26	24	20	16	80000	34	32	26	24
40000	26	24	20	18	90000	38	37	27	26
50000	29	28	24	20	100000	45	42	28	29

Table 5.3: Impact of threshold value on relaxed selection for relation *in*.



(a) Time (b) Visited Nodes (c) Compatible Features

Figure 5.7: Definable selection conditions.

5.3.1.5 Definable Selection Conditions

This group of experiments aims at analyzing performance of IBF_θ, ITHR(v_{no_relax})_θ, (1 in our case), ITHR(v_3)_θ, (0.8 in our case) and INN_θ, in executing definable selection conditions.

Figure 5.7 reports the results concerning the execution of a set of topological selection queries with respect to the same query object. For each query, the set of the well defined relations corresponding to the definable query relation are pointed out, using notation introduced in Table 3.1 in Section 3.2. As an example, we report the results obtained executing queries against the dataset of size 50000.

As we can see, different relations have different impact on query performance. As expected, by increasing the number of relations considered in a computation, time and compatible features increase as well. In particular, it is possible to identify two different groups of relations. One contains only high selective relations, such as *coveredby*, *in*, *contains*, *covers*; notice that, for the threshold v_3 (0.8 in our case) such group does not consider

in the relaxation process some low selective relations such as *overlap*. The other contains also low selective relations such as *overlap*. The first group has best performance and consider a low number of compatible features, while as expected the second group has an higher response time since an higher number of compatible features are considered. On the other side, the percentage of nodes almost coincide for all the techniques on all the considered queries. This is due to the following facts: (i) the percentage of accessed nodes depends on the considered compatibility relation (see Table 4.4); (ii) for intermediate nodes of the R*-tree, the sets of compatible relations are quite similar for all topological relations we consider (see Table 4.4). Therefore, the pruning strategy does not vary too much by changing the query predicate.

In general, response times of the considered techniques are very similar to the corresponding experiments on well-defined relations (Figures 5.6), confirming that the definable selection conditions require a negligible additional amount of computation.

5.3.2 Results for Gaussian and Skewed Distribution

The aim of the experiments with non-uniform distributions of datasets is to highlight the impact of the distribution of the considered datasets on the proposed techniques. We distinguish two different positions of query objects for Gaussian distribution and three different positions for skewed distribution. More precisely, we consider three different query types depending on the position of the query objects:

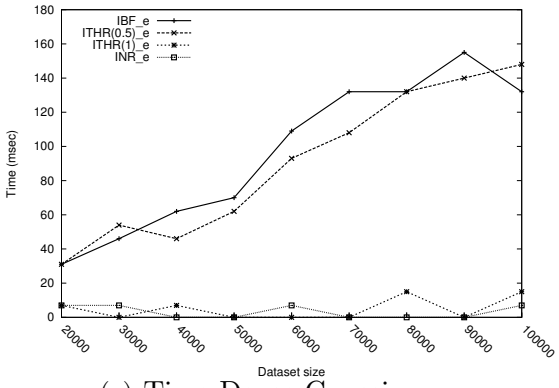
- *Dense queries.* A query is dense if the query object belongs to a dense area of the space. For Gaussian distribution, a query is dense if the query object is located in the center of the global area; for skewed distribution a query is dense if the query object is located in the high left corner.
- *Non dense queries.* A query is non dense if the query object belongs to a non dense area of the space. For Gaussian distribution, a query is non dense if the query object is located in the border of the global area. For skewed a query is non dense if the query object is located in the low right corner.
- *Semi dense queries.* A query is semi dense if the query object belongs to a medium dense area with of the space. Only for skewed a query is non dense if the query object is located in the diagonal from the low left corner to the high right corner.

For Gaussian distribution the 10 executions are divided into: 2 executions of dense queries and 8 execution of non dense queries; for skewed distribution the 10 executions are divided into: 2 executions of dense queries, 6 executions of semi dense queries, and 2 executions of non dense queries. For each distribution we separate the types of query and for each types

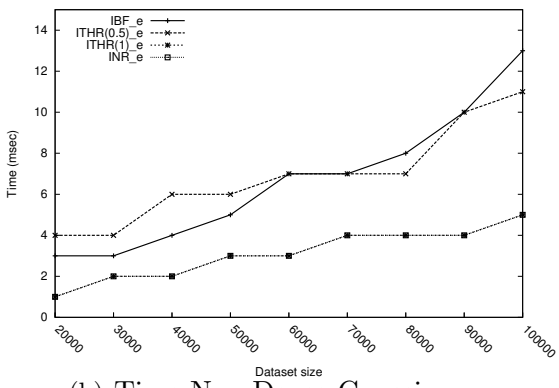
of query as final result we consider the average of the statistics resulting from the group of executions.

In this section we report the results obtained for datasets with Gaussian and skewed distributions.

5.3.2.1 Relaxed Methods with respect to Non Relaxed One

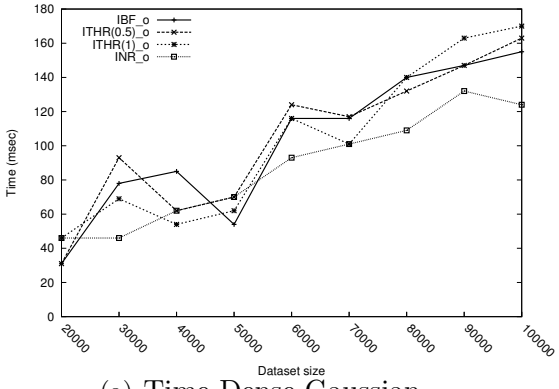


(a) Time Dense Gaussian

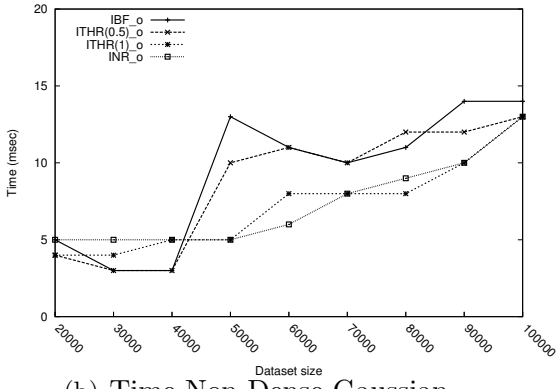


(b) Time Non Dense Gaussian

Figure 5.8: Relaxed methods with respect to non relaxed ones for *equal* (high selective relation) for gaussian distribution.



(a) Time Dense Gaussian

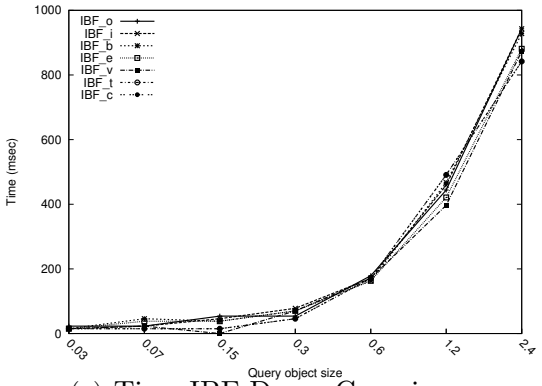


(b) Time Non Dense Gaussian

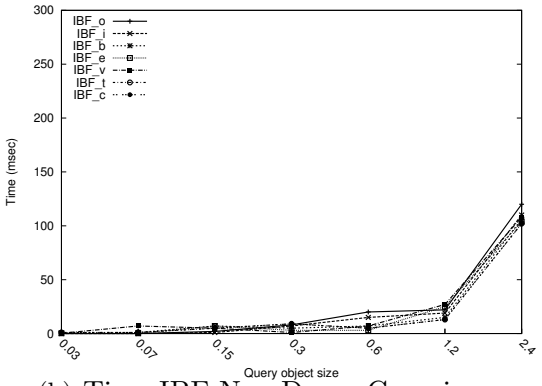
Figure 5.9: Relaxed methods with respect to non relaxed ones for *overlap* (low selective relation) for Gaussian distribution.

The main consideration is that all the different types of queries, considering both Gaussian and skewed distribution have a similar behaviour.

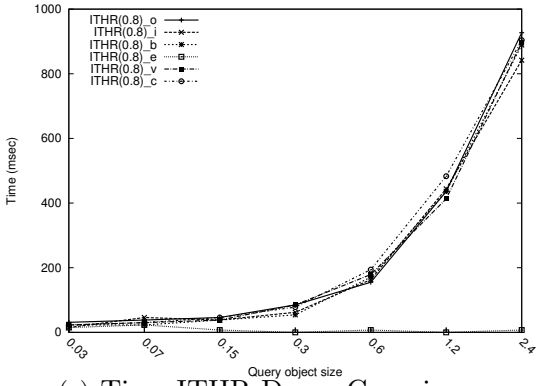
In this group we compare IBF_θ, ITHR_θ,, INR_θ, by considering as threshold values v_{no_relax} (1 our case), and v_{no_dj} (0.5 in our case), a high selective relation, *equal*, and a



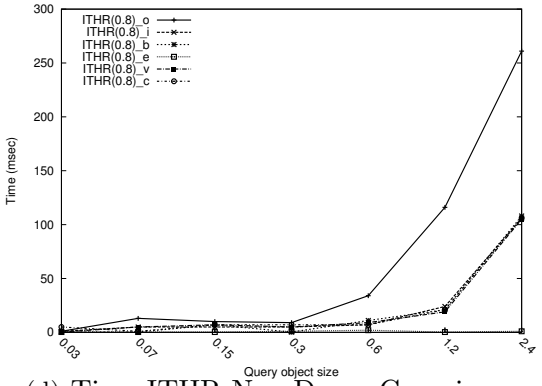
(a) Time IBF Dense Gaussian



(b) Time IBF Non Dense Gaussian



(c) Time ITHR Dense Gaussian



(d) Time ITHR Non Dense Gaussian

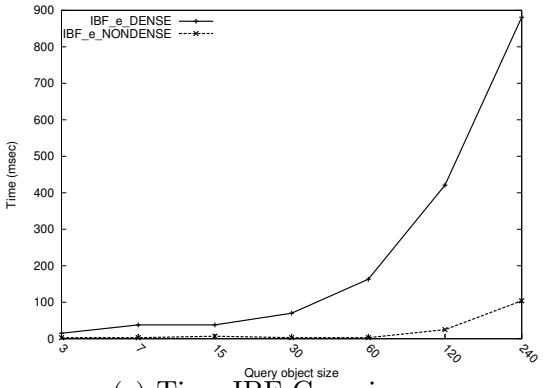
Figure 5.10: Impact of the query object size.

low selective relation *overlap*. The criteria we use for these experiments are the same considered for uniform distribution (see Section 5.3).

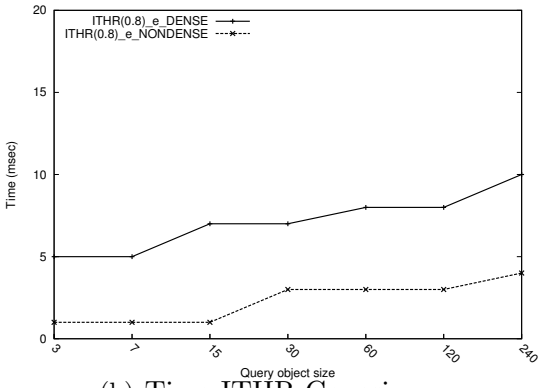
Figures 5.8 and 5.9 show the response time for dense and non dense queries for Gaussian distribution considering respectively high selective relation *equal* and low selective relation *overlap*.

On the other side, Figures 5.13 and 5.14 show the response time for dense and non dense queries for skewed distribution considering respectively high selective relation *equal* and low selective relation *overlap*.

Considering the high selective relation for both Gaussian and skewed distributions relaxed techniques are less efficient than non relaxed ones. Thus as a consequence ITHR(v_{no_relax})_θ and INR_θ for *equal* have a lower response time with respect to ITHR(v_{no_dj})_θ and IBF_θ considering dense and non dense queries for Gaussian (see Figure 5.8) and dense, semi dense and non dense queries for skewed distribution (see Figure 5.13). Considering low selective relation for both Gaussian and skewed distributions there are not relevant differences between relaxed and non relaxed techniques, since for low selective predicates the

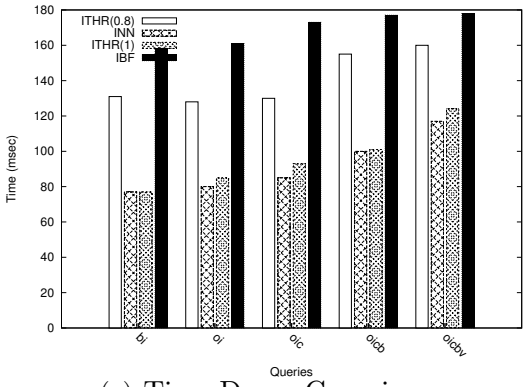


(a) Time IBF Gaussian

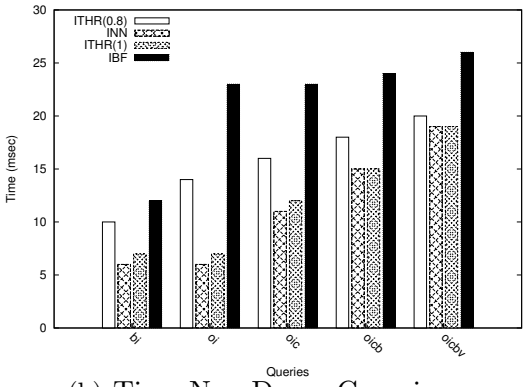


(b) Time ITHR Gaussian

Figure 5.11: Impact of the query object size.



(a) Time Dense Gaussian



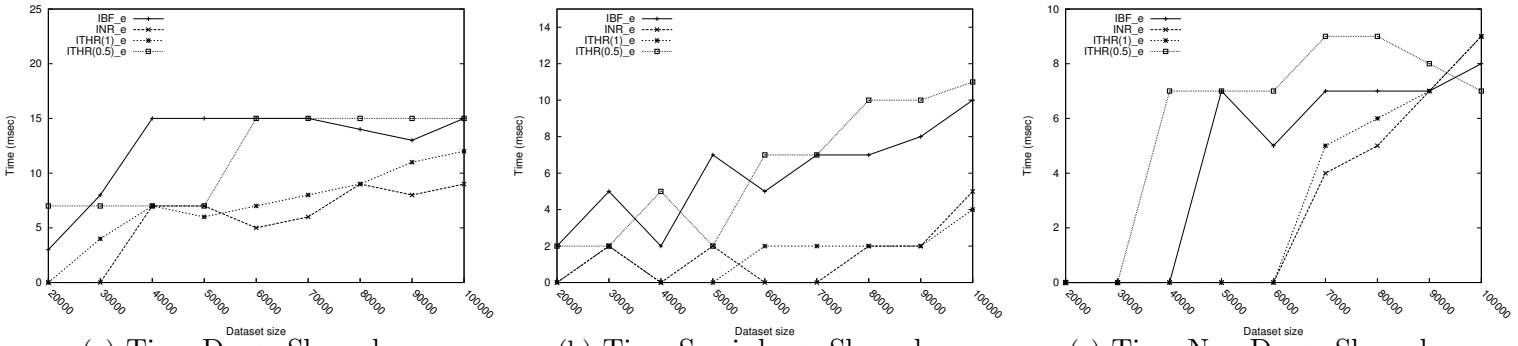
(b) Time Non Dense Gaussian

Figure 5.12: Definable selection conditions.

relaxation process does not increase the dimension of the results.

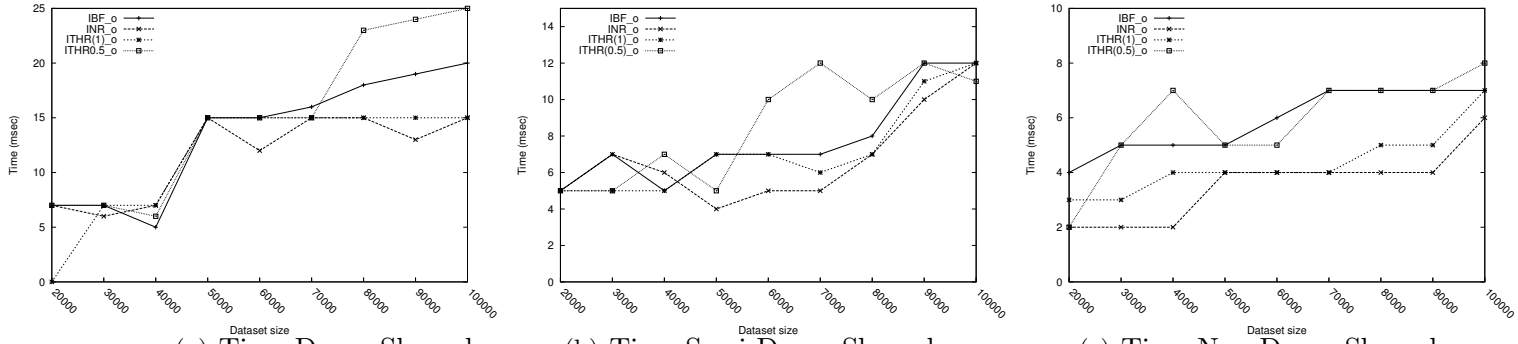
In this group of experiments the differences between Gaussian and skewed depends on the characteristics of the distribution, in fact, skewed perform better than Gaussian since skewed is more grouped thus it is not frequent that the query object intersects a great number of MBRs of the skewed dataset. This is more evident for semi dense queries over skewed distribution, in fact, there is a further decreasing of the percentage of features and nodes considered and a consequent response time improvement.

On the other side considering the three distributions, uniform, Gaussian and skewed, Gaussian has the highest response time, this is due to the fact that the positions of the query objects for dense query over Gaussian distribution are suitable to intersect a large part of the MBRs of the datasets.



(a) Time Dense Skewed (b) Time Semi dense Skewed (c) Time Non Dense Skewed

Figure 5.13: Relaxed methods with respect to non relaxed ones for *equal* (high selective relation) for skewed distribution.



(a) Time Dense Skewed (b) Time Semi Dense Skewed (c) Time Non Dense Skewed

Figure 5.14: Relaxed methods with respect to non relaxed ones for *overlap* (low selective relation) for skewed distribution.

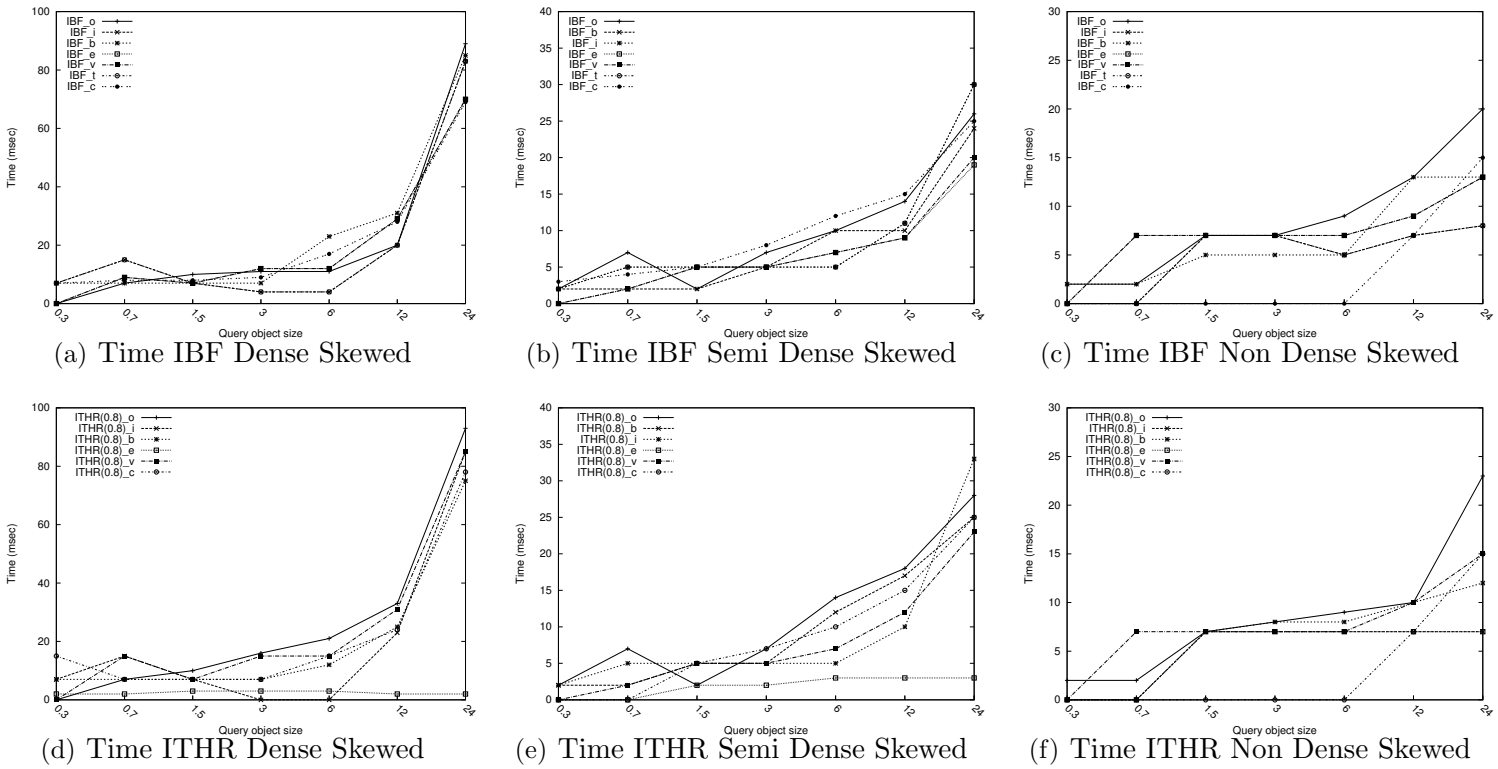
5.3.2.2 Impact of the Query Object Size

For this group of experiments ,we consider IBF_θ, ITHR(v_3)_θ (ITHR(0.8)_θ) with the purpose of analyzing the impact of query object size on query selectivity for relations, excluding *disjoint* and *touch* for ITHR(0.8)_θ and only *disjoint* for IBF_θ.

Figures 5.10, 5.15 show the response time of IBF_θ and ITHR(v_3)_θ techniques for dense and non dense queries over Gaussian distribution and over dense, semi dense and non dense for skewed distribution. As expected, response time increases while increasing object size, since selectivity decrease, thus, more nodes and objects are considered. Moreover, the response time for non dense queries is significantly lower than that for dense queries for both Gaussian distribution and than dense and semi dense for skewed distributions.

Considering dense queries, the difference between Gaussian, skewed and uniform concern the decreasing of performance of all algorithms for the skewed and uniform distribution w.r.t Gaussian distribution. In fact the response time for Gaussian distribution, for IBF_θ and ITHR_θ, is almost double w.r.t the response time of uniform and about 10 times the response time of skewed.

For IBF_θ semantics there are not relevant differences among relations in response time since all considered relations are probably relaxed into *overlap*, that is the second more frequent relations after *disjoint*. On the other hand, for ITHR_θ semantics since the num-



(a) Time IBF Dense Skewed (b) Time IBF Semi Dense Skewed (c) Time IBF Non Dense Skewed

(d) Time ITHR Dense Skewed (e) Time ITHR Semi Dense Skewed (f) Time ITHR Non Dense Skewed

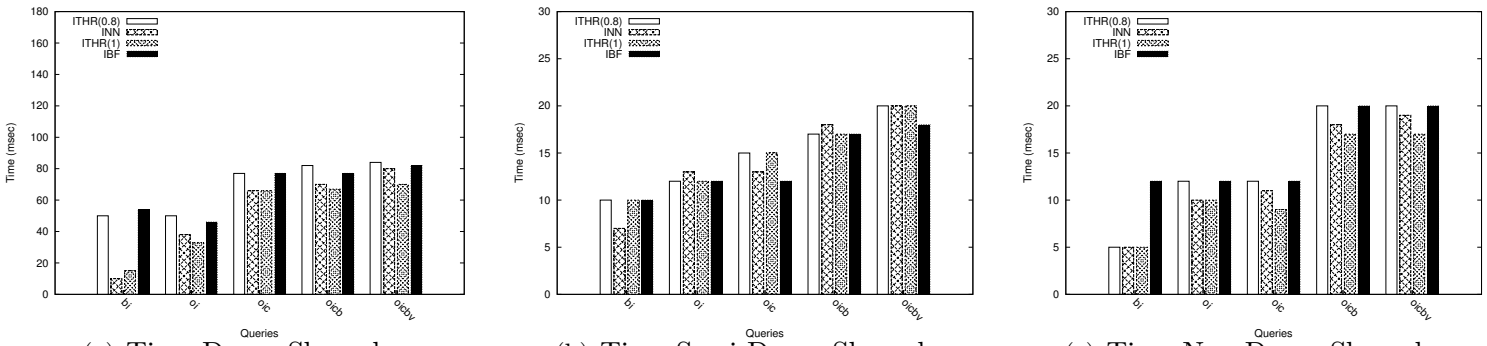
Figure 5.15: Impact of the query object size.

ber of relations in the relaxation depends on the considered threshold value, the difference among relations is evident. In fact, since *equal* is a very selective relation and since for threshold value 0.8 *equal* relaxes into *equal* itself, its response time is lower than thats of other relations.

In general, comparing skewed and Gaussian distributions, we notice that skewed distribution has a very small response time with respect to Gaussian distribution. Since the characteristics of Gaussian distribution are more similar to uniform distribution than to skewed distribution, the obtained results of Gaussian distribution are more similar to those obtained for uniform distribution

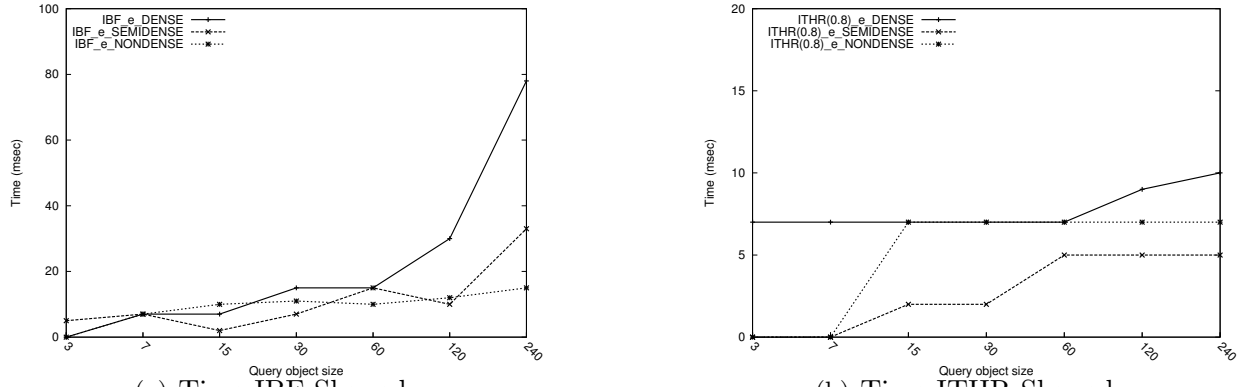
Figures 5.11, 5.17 shows the impact of the query types on the performance of IBF and ITHR(v_{no_dj})_θ, considering an high selective predicate, such as *equal*. In the figures, we use the following notations:

- IBF_e_DENSE, ITHR(0.8)_e_DENSE: the response time for obtained for dense queries;
- IBF_e_NONDENSE, ITHR(0.8)_e_NONDENSE: the response time for non dense queries;
- IBF_e_SEMIDENSE, ITHR(0.8)_e_SEMIDENSE: the response time for semi dense



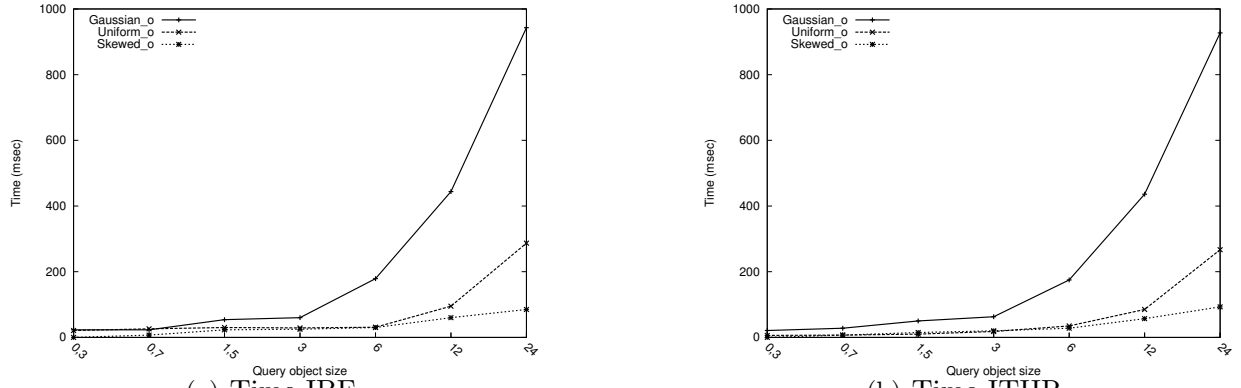
(a) Time Dense Skewed (b) Time Semi Dense Skewed (c) Time Non Dense Skewed

Figure 5.16: Definable selection conditions.



(a) Time IBF Skewed (b) Time ITHR Skewed

Figure 5.17: Impact of the query object size.



(a) Time IBF (b) Time ITHR

Figure 5.18: Best Fit and Threshold performance with respect to query object size for dense query for all types of distribution.

queries.

As expected, dense queries have the high response time for both Gaussian and skewed

distributions, since the dimension of the results is higher than non dense ones; moreover ITHR_θ performs in general better to IBF_θ , since for the threshold value v_{no_dj} the relation *equal* relaxes into *equal* itself.

Figure 5.18 shows the comparison between uniform, Gaussian and skewed distribution, denote respectively by Uniform_o, Gaussian_o and Skewed_o for IBF_θ and ITHR(v_{no_dj})_θ, considering a low selective relation, such as *overlap*. Gaussian has the worst performance for both techniques due to the fact that the density of Gaussian is higher w.r.t skewed and uniform thus the central objects considered may intersect a very high number of objects.

5.3.2.3 Impact of Threshold Value

For this group of experiments we consider the response time in msec for ITHR(ρ)_θ taking into account the four threshold values and relation *in*, as for the uniform distribution.

By analyzing Tables 5.4, 5.5, 5.6, 5.7, 5.8 we can observe that performance improves by increasing the Threshold value since less relations are considered in the processing. Moreover, when considering query type we notice that, as expected, the response time for non dense queries is better than response time for dense queries since in the first case less objects are taken into account.

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
20000	3	3	3	2	70000	7	7	5	5
30000	3	5	3	2	80000	11	9	7	6
40000	3	5	3	3	90000	13	10	7	6
50000	9	5	5	5	100000	15	11	9	8

Table 5.4: Impact of threshold values for non dense query in Gaussian distribution.

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
20000	62	51	22	18	70000	208	180	59	58
30000	101	77	32	39	80000	216	208	65	61
40000	112	87	39	36	90000	240	220	69	63
50000	153	103	49	48	100000	268	250	74	62

Table 5.5: Impact of threshold value for dense query in Gaussian distribution.

5.3.2.4 Definable Selection Conditions

In this group of experiments, we analyze the techniques proposed for definable conditions: IBF_θ, ITHR(v_{no_relax})_θ (1 in our case), and ITHR(v_3)_θ (0.8 in our case). The presented

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
20000	38	30	11	10	70000	109	101	55	53
30000	47	38	20	18	80000	148	116	67	60
40000	57	52	38	32	90000	150	145	80	77
50000	72	68	43	40	100000	179	163	120	108

Table 5.6: Impact of threshold value for dense query in skewed distribution.

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
20000	6	5	0	0	70000	22	12	8	7
30000	7	7	7	7	80000	23	13	9	7
40000	7	7	7	7	90000	22	22	15	13
50000	12	7	7	7	100000	23	28	15	15

Table 5.7: Impact of threshold values for semi dense query in skewed distribution.

Response time in msec									
Dataset size	**0.5**	**0.6**	**0.8**	**1**	**Dataset size**	**0.5**	**0.6**	**0.8**	**1**
20000	5	5	0	0	70000	10	7	8	0
30000	5	7	0	0	80000	11	10	7	0
40000	5	7	0	0	90000	13	11	7	7
50000	9	7	0	0	100000	14	12	7	7

Table 5.8: Impact of threshold values for non dense query in skewed distribution.

results refer to the dataset of size 50000.

Figure 5.12 shows response time for dense and non dense queries for Gaussian distribution and Figure 5.16 shows response time for dense, semi dense, and non dense queries for skewed distribution.

We consider the following groups of relations: *bi*, *oi*, *oic*, *oicb*, *oicbv*. As expected, when the *overlap* relations is considered the response time increase for all techniques, due to the higher percentage of compatible features considered. In fact, the group with the lowest response time is *bi* since for the threshold value 0.8 both relation b and i do not consider overlap in the relaxation, and also for IBF it is possible that both b and i are relaxed into a relation different from *overlap*.

As expected, also in this case skewed performs better than Gaussian for all types of queries. Moreover non dense queries have similar results to uniform distribution that is a lower response time.

5.4 Results for Relaxed Join Operator

In order to evaluate the performances of the proposed algorithms for relaxed join, we consider four different cases, corresponding to different distributions of the input datasets. The main purpose of such experiments is to analyze performances with respect to the variation of the distribution of the objects in the datasets and of the rate of overlap of the two datasets. We consider two different MBR sizes, 12 and 30 units respectively, corresponding to the 0.1 %, 0.3 % of the global area.

For the cases *uniform-skewed*, we perform a spatial join between two datasets with the same cardinality: the first dataset has a uniform distribution, while the second has a skewed distribution. We have then executed three groups of experiments. The aim of the first group of experiments is to compare the performance of indexed relaxed selection with respect to the execution of non relaxed selection, in order to analyze the impact of query relaxation. The second group of experiments aims at analyzing the impact of size of MBRs datasets on query selectivity. The third group of experiments aims at analyzing the impact of the threshold value on performance.

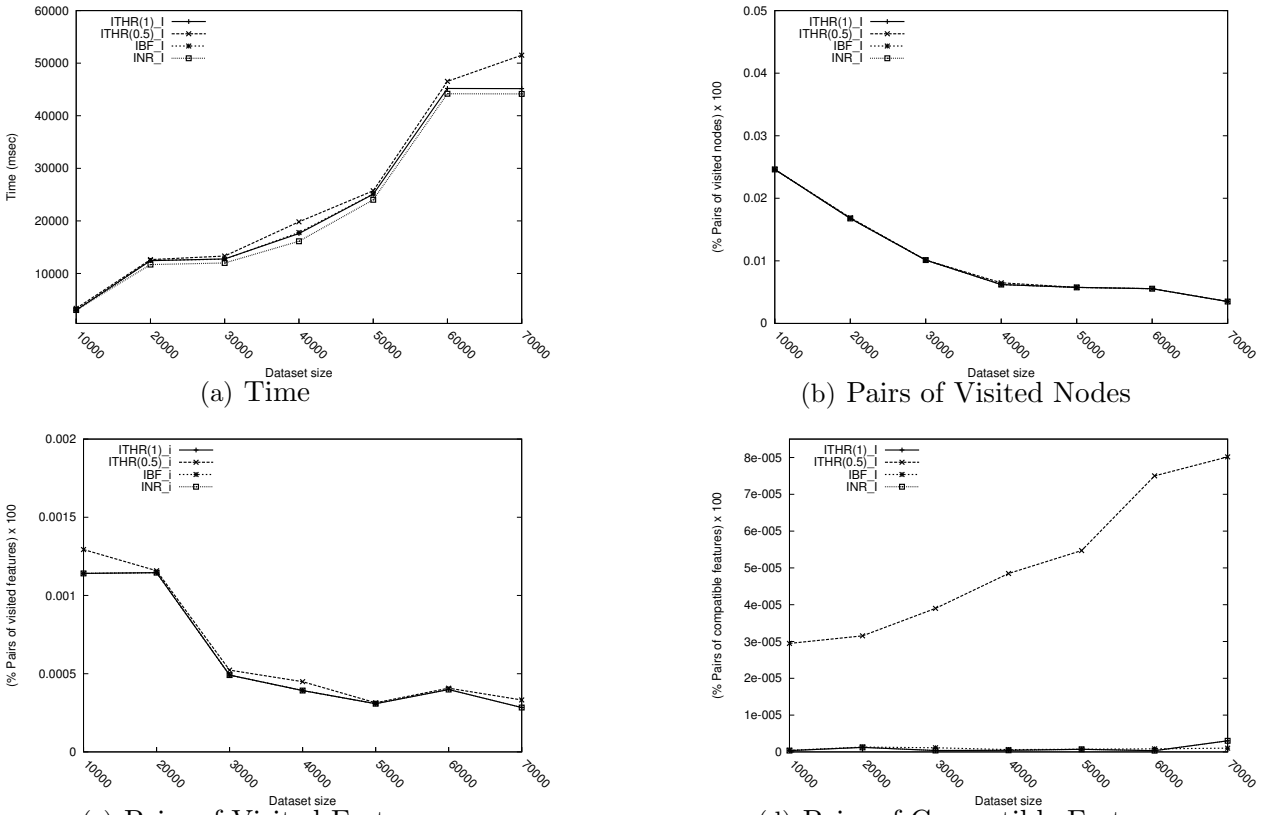
For the cases *uniform-Gaussian*, we perform a spatial join between two datasets with the same cardinality, the first dataset has a uniform distribution, while the second has a Gaussian distribution. We have then executed three groups of experiments. The aim of the first group of experiments is to compare the performance of indexed relaxed selection with respect to the execution of non relaxed selection, in order to analyze the impact of query relaxation. The second group of experiments aims at analyzing the impact of size of MBRs datasets on query selectivity. The third group of experiments aims at analyzing the impact of the threshold value on performance.

For the case *uniform-uniform* we perform a relaxed join over two datasets obtained by translating a given dataset on the right bottom corner in order to change the rate of overlap. The aim of this group of experiment is to highlight the impact of different overlap rates between the two datasets on the proposed algorithms.

Finally for the case *skewed-Gaussian*, we perform a spatial join between two datasets with the same cardinality, the first dataset has a skewed distribution, while the second has a Gaussian or a skewed distribution. In this case, the aim of the experiment is to analyze the impact of the number of intersections on the performance.

5.4.1 Case Uniform-Skewed

For this case we perform groups of experiments similar to those already discussed for the relaxed selection operator (see Section 5.3.2).



(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

Figure 5.19: Relaxed methods with respect to non relaxed ones over uniform-skewed datasets MBR size ≤ 12, relation *in* (high selective relation).

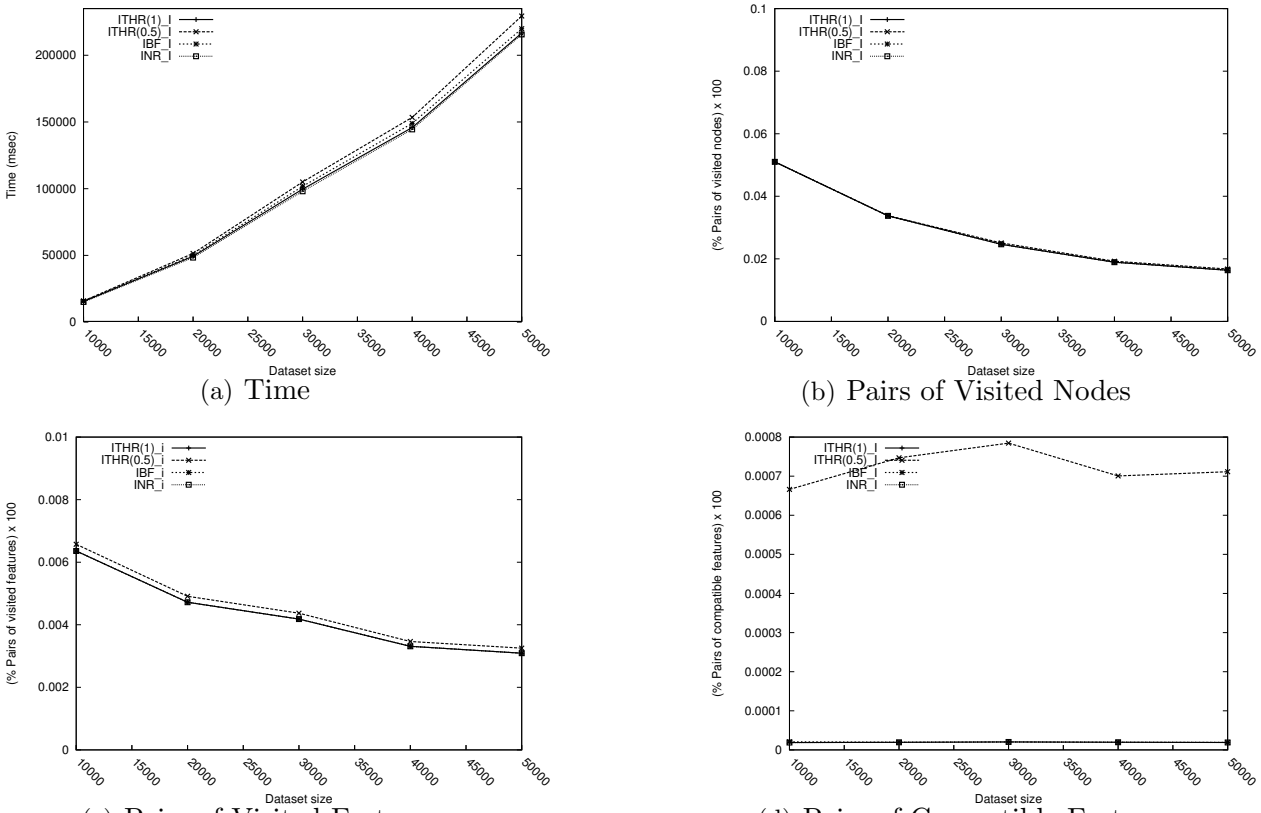
5.4.1.1 Relaxed Methods with respect to Non Relaxed Ones

The aim of this group of experiments is to compare IBF_θ, ITHR_θ, and INR_θ. As threshold values, we consider v_{no_relax} (1 in our case) and v_{no_dj} (0.5 in our case), which is the similarity value which excludes $disjoint$ from the set of similar relations for all topological relations excluding $disjoint$ and $touch$. Notice that, as pointed out for selection, ITHR(1)_θ and INR_θ always return similar results.

We chose the value v_{no_dj} in order to consider the worst case for relaxation to evaluate how the maximum relaxation impact to the performance of the presented techniques. On the other side we consider v_{no_relax} in order to evaluate the additional costs of a relaxed execution w.r.t. non relaxed execution.

We consider a high selective relation such as *in* [3] and a low selective relation such as

[3]We consider *in* instead of *equal*, as performed for selection, since *equal* is a too strong selective relation, thus it is not suitable to highlight the differences between the proposed techniques.



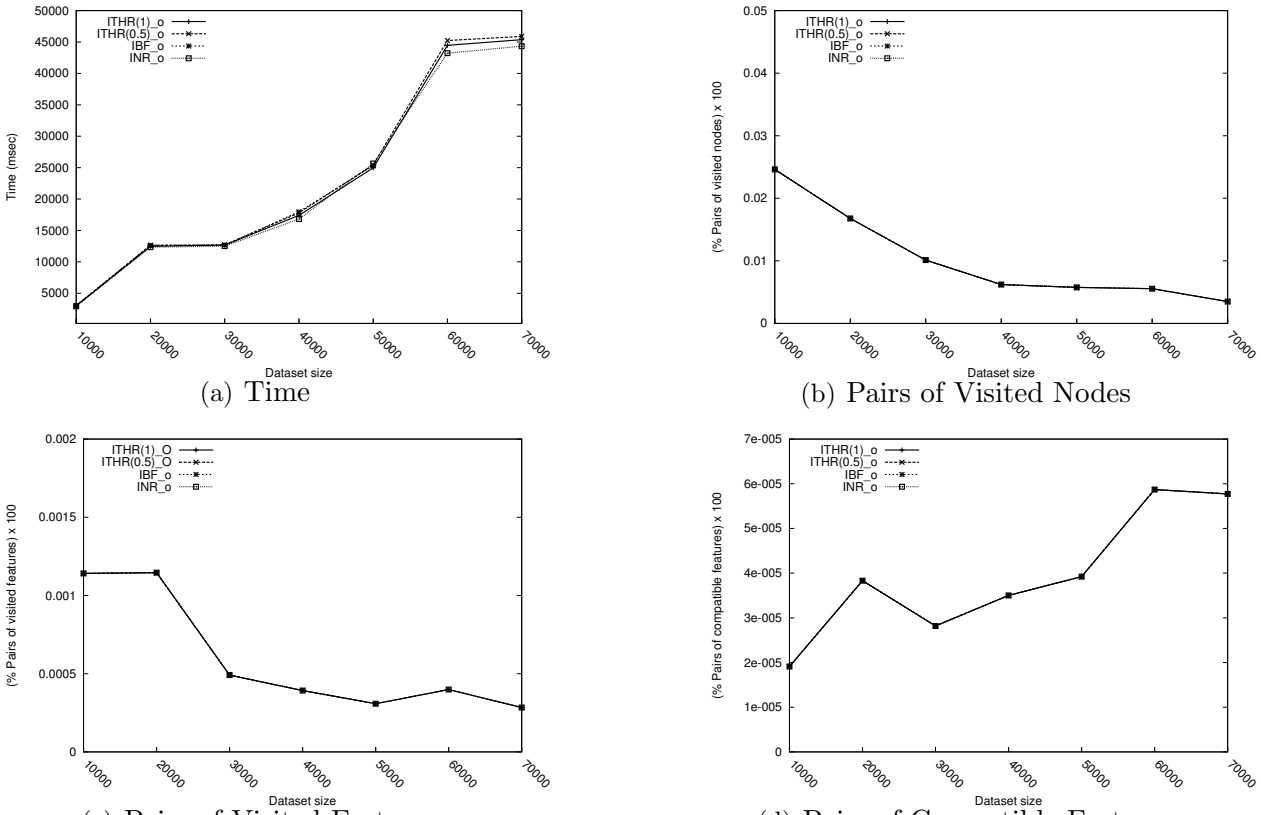
(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

Figure 5.20: Relaxed methods with respect to non relaxed ones over uniform-skewed datasets MBR size ≤ 30, relation *in* (high selective relation).

overlap. In this case *in* is high selective relation since the MBRs of the two datasets joined have the similar dimensions.

Figures 5.19 and 5.20 report results concerning relation *in*. We can see that results obtained for the various techniques are quite similar, with a slightly higher response time of ITHR(0.5)_θ. This is due to the percentage of compatible features, that is higher for ITHR(0.5), since relaxing *in* we consider also low selective relations such as *overlap*. On the other side, the result for the visited nodes are the quite the same for IBF_θ, ITHR(1)_θ, and INR_θ, while the percentage of visited features is slightly higher for ITHR(0.5)_θ. This is due to the following facts: (i) the percentage of accessed nodes depends on the considered compatibility relation (see Table 4.5); (ii) for intermediate nodes of the R*-tree, the sets of compatible relations are quite similar for all topological relations we consider (see Table 4.5). Therefore, the pruning strategy does not vary too much by changing the query predicate. As expected the response time of size 30 is higher than response time of size 12 and also the percentage of visited nodes, visited features and compatible features is higher than size 12. This is due to the fact that the density for size 30 is higher than density for



(a) Time

(b) Pairs of Visited Nodes

(c) Pairs of Visited Features

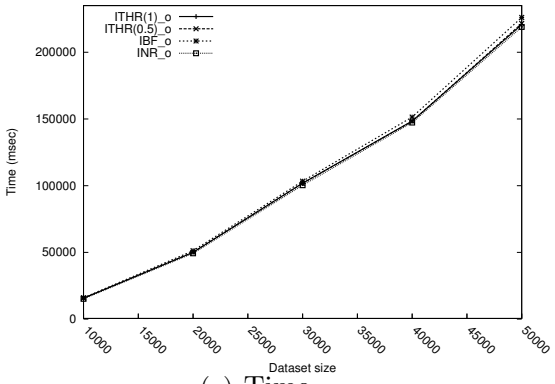
(d) Pairs of Compatible Features

Figure 5.21: Relaxed methods with respect to non relaxed ones over uniform-skewed datasets MBR size ≤ 12, relation *overlap* (low selective relation).

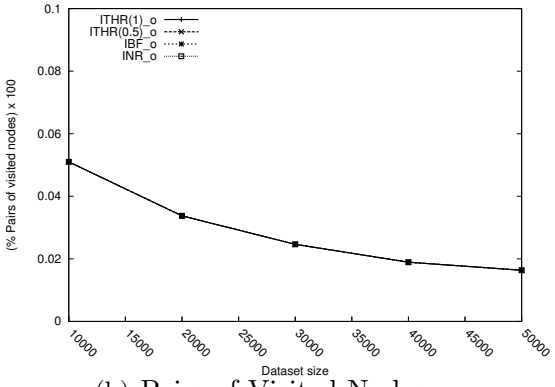
size 12.

Figures 5.21 and 5.22 report results for a low selective relation, *overlap*. For both size 12 and 30 response time, percentage of visited nodes and compatible features are similar for all the considered techniques. Indeed, in this case: (i) it is highly probable that objects satisfying the query predicate exist, thus IBF_θ behaves like INR_θ; (ii) usually all predicates but disjoint are more selective than *overlap*, ITHR(1)_θ has a similar trend to INR_θ, since new relations considered by the relaxation do not significantly alter the result set. On the other side, the response time, the percentage of visited nodes, visited features and compatible features are higher for size 30 w.r.t size 12.

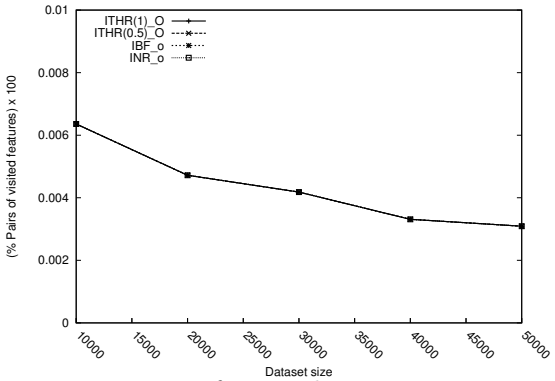
Comparing low and high selective relations, notice that the main differences are for response time and compatible features that are both higher for ITHR(0.5)_θ for the high selective relation *in*, since it in the relaxation process it considers more relations than *overlap*.



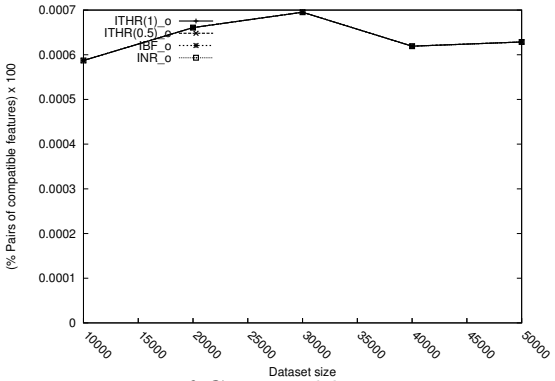
(a) Time



(b) Pairs of Visited Nodes



(c) Pairs of Visited Features



(d) Pairs of Compatible Features

Figure 5.22: Relaxed methods with respect to non relaxed ones over uniform-skewed datasets MBR size ≤ 30, relation *overlap* (low selective relation).

5.4.1.2 Impact of the MBR Size

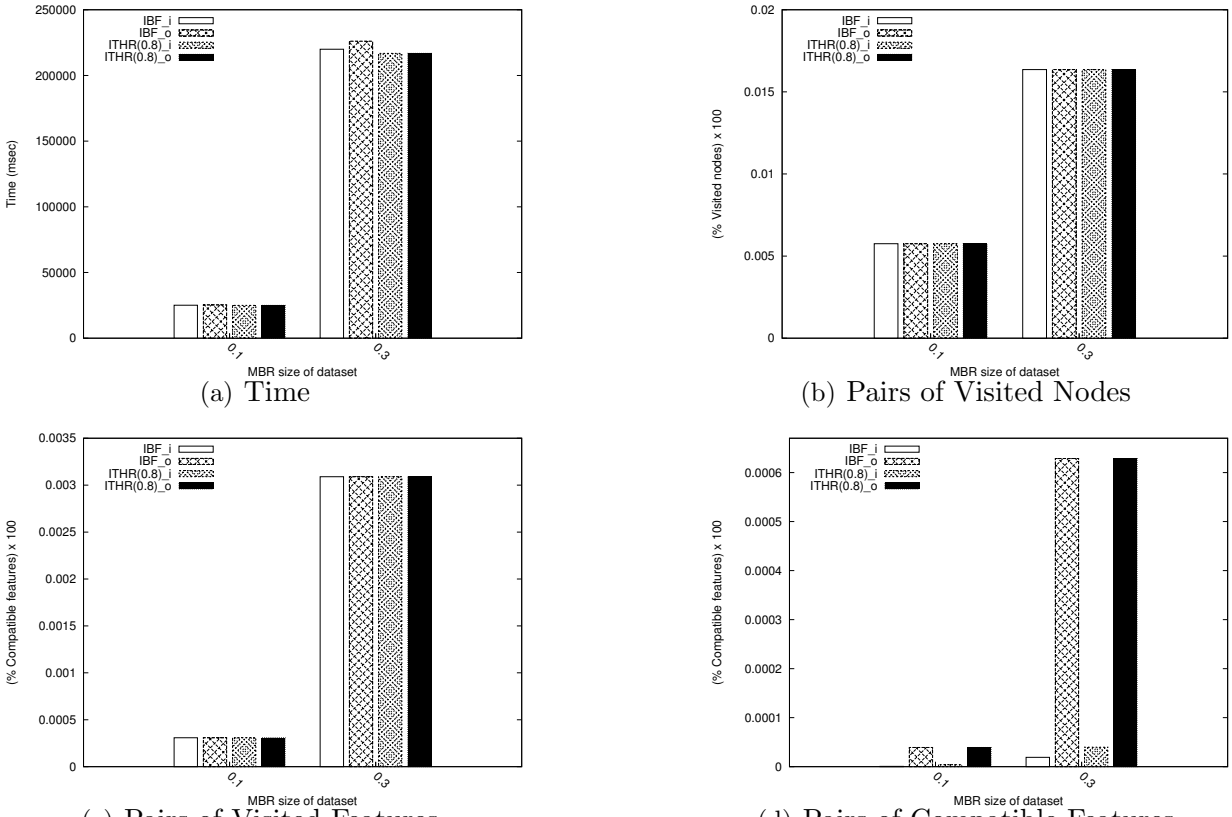
The techniques considered in this group of experiments are IBF_θ and ITHR(v_3)_θ (ITHR(0.8)_θ in our case) in order to analyze the impact of the MBR size of datasets for both a high selective (*in*) and a low selective (*overlap*) relations.

As we can see from Figures 5.23 for the considered techniques, the response time, the percentage of nodes, visited features and visited nodes grow up while increasing the MBR size, since MBR density is higher. Moreover, as expected the statistics of the low selective relation *overlap* are higher than the high selective relation *in*, considering in particular the compatible features.

This is due to the following facts: (i) the number of objects satisfying *overlap* is significantly higher than the number of objects satisfying *in*; (ii) in executing ITHR(0.8)_i no low selective relations, such as *overlap*, are taken into account by the relaxed evaluation.

On the other side, for visited nodes and features there are no difference due to the definition

of the compatibility relation considered (see Table 4.5).



(a) Time

(b) Pairs of Visited Nodes

(c) Pairs of Visited Features

(d) Pairs of Compatible Features

Figure 5.23: Impact of the datasets MBR size for uniform-skewed distributions.

5.4.1.3 Impact of Threshold Value

The aim of these experiments is to evaluate the impact of the threshold value on the technique ITHR_θ. We consider the response time in msec of ITHR(ρ)_θ by considering four threshold values ($v_{no\_relax}, v_{P2}, v_3, v_{no\_dj}$), considering only the MBR dataset dimension 30. Also in this case, the relation *in* is considered for its characteristics already discussed for the selection query.

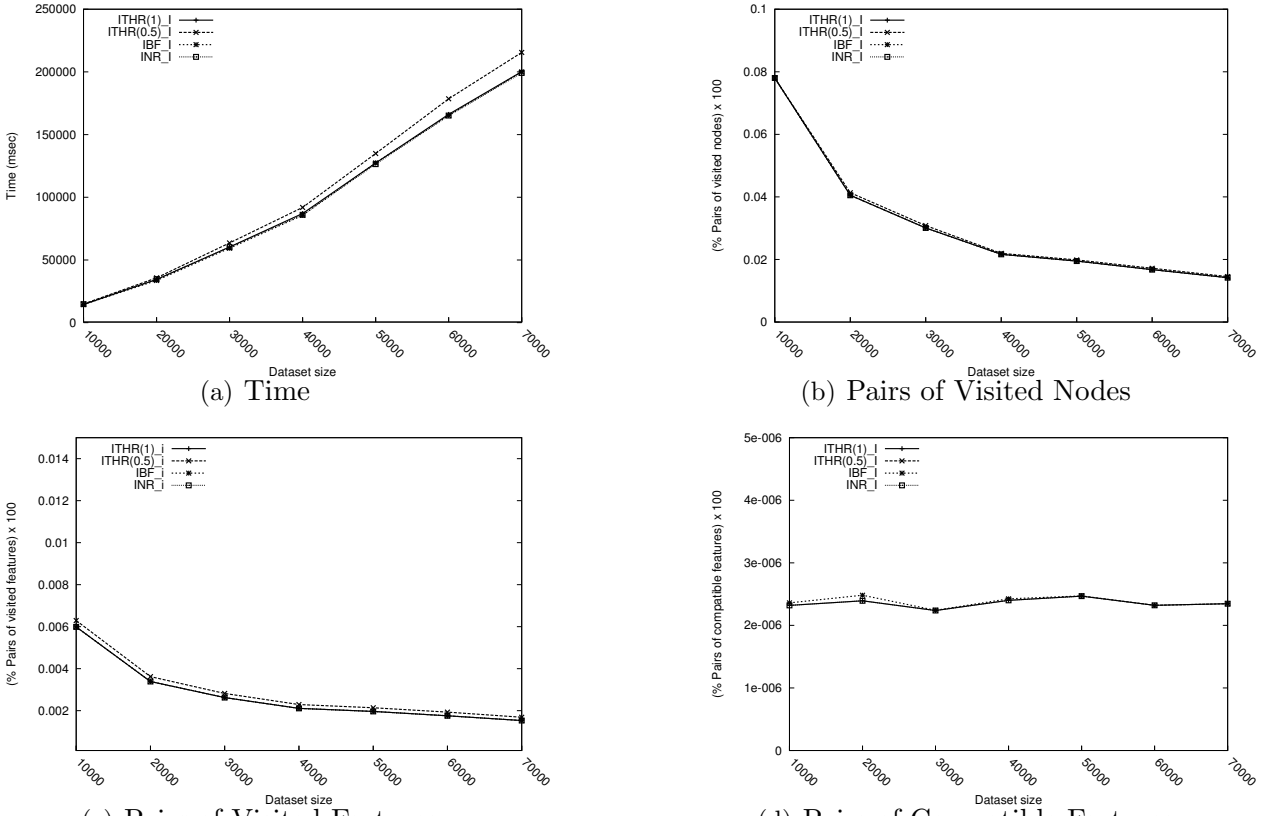
As expected, by increasing the threshold value performance improves since less relations are taken into account during the search, as highlighted in Table 5.9.

5.4.2 Case Uniform-Gaussian

The sets of experiments presented are the same presented for uniform-skewed distribution.

Response time in msec				
Dataset size	**0.5**	**0.6**	**0.8**	**1**
10000	15849	15591	15428	15475
20000	51464	50060	49444	49280
30000	105027	101525	99848	99676
40000	153465	148309	145688	145813
50000	229422	220967	216802	216646

Table 5.9: Impact of threshold value for uniform-skewed distributions.



(a) Time (b) Pairs of Visited Nodes

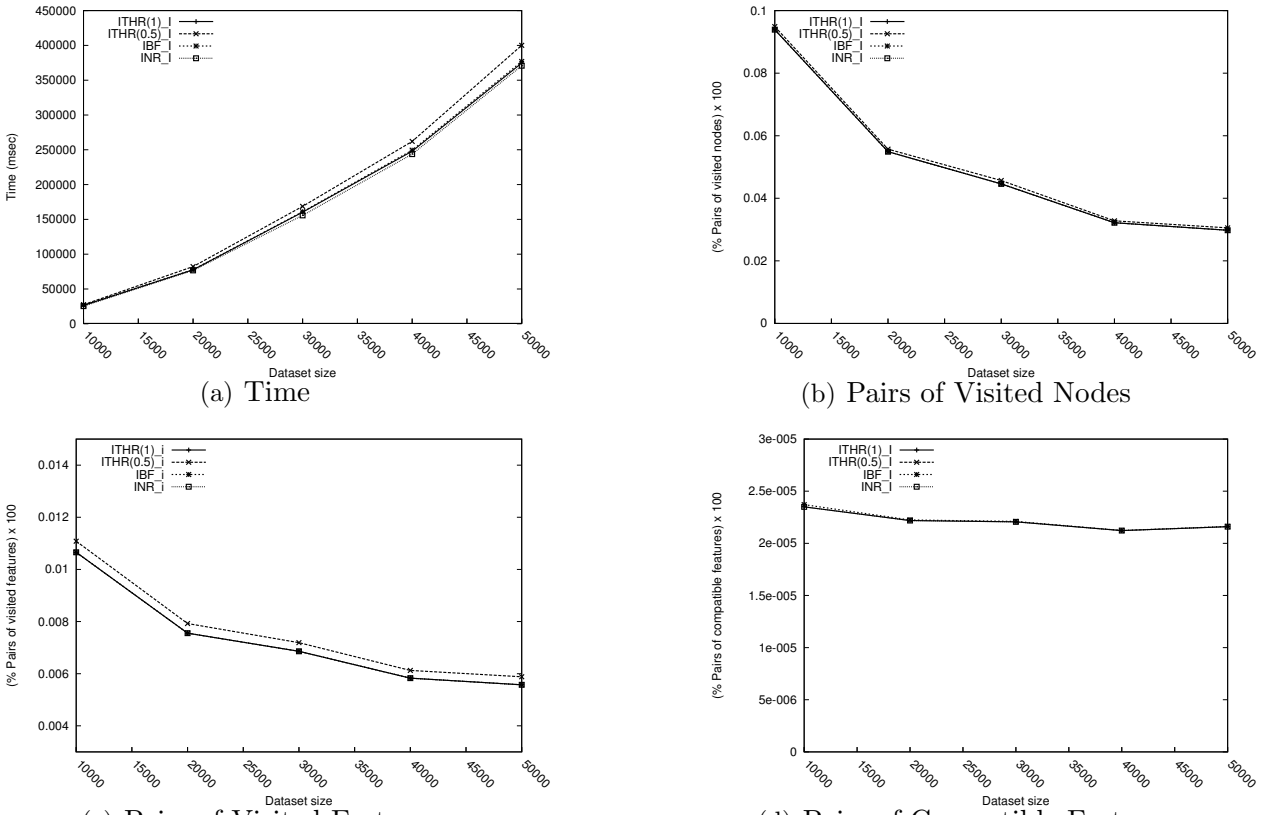
(c) Pairs of Visited Features (d) Pairs of Compatible Features

Figure 5.24: Relaxed methods with respect to non relaxed ones over uniform-Gaussian datasets MBR size ≤ 12, relation *in* (high selective relation).

5.4.2.1 Relaxed Methods with respect to Non Relaxed Ones

The aim of this group of experiments is to compare IBF_θ, ITHR_θ, and INR_θ. In this case we adopt the threshold values considered for the case uniform-skewed.

Figures 5.24 and 5.25 report results concerning relation *in*. Also for uniform-Gaussian, there is a difference in the response time, in particular the relaxed technique ITHR(v_{no_dj})_θ



(a) Time (b) Pairs of Visited Nodes

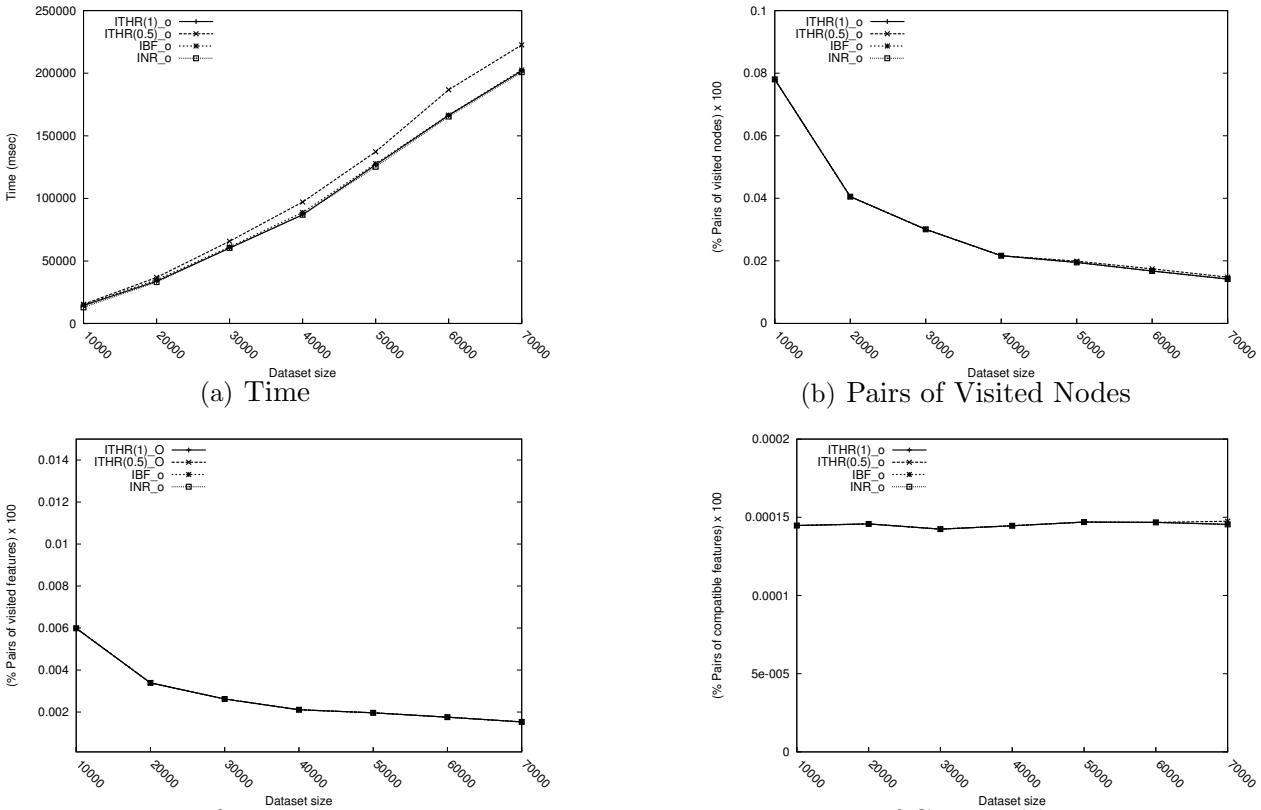
(c) Pairs of Visited Features (d) Pairs of Compatible Features

Figure 5.25: Relaxed methods with respect to non relaxed ones over uniform-Gaussian datasets MBR size ≤ 30, relation *in* (high selective relation).

has the highest response time. This is due to the percentage of visited features, that is higher for ITHR(v_{no_dj})_θ. On the other side, the result for the response time, the visited nodes and the compatible features are the same for IBF_θ, ITHR(v_{no_relax})_θ, and INR, the reasons of such behaviour are the same explained for the case uniform-skewed.

Moreover, also in this case, the response time of size 30 is higher than response time of size 12 and also the percentage of visited nodes, visited features and compatible features is higher than size 12. This is due to the fact that the density for size 30 is higher than density for size 12.

Figures 5.26 and 5.27 show results for a low selective relation, *overlap*. Also the behaviour for *overlap* for uniform-Gaussian is similar to those for uniform-skewed, thus, for both size 12 and 30 response time, percentage of visited nodes and compatible features have similar trend for all the considered techniques, due to the low selectivity of the predicate. Moreover, the response time, the percentage of visited nodes, visited features and compatible features are higher for size 30 w.r.t size 12.



(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

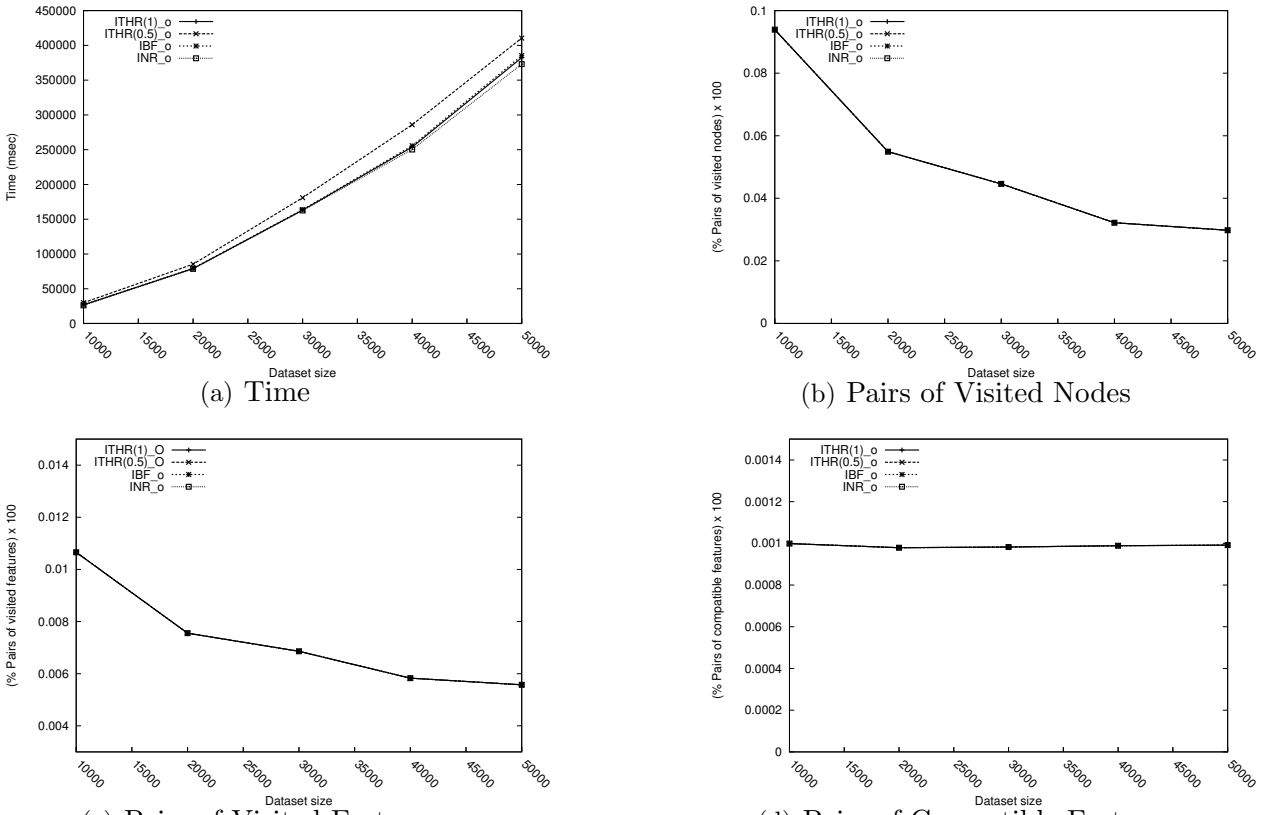
Figure 5.26: Relaxed methods with respect to non relaxed ones over uniform-Gaussian datasets MBR size ≤ 12, relation *overlap* (low selective relation).

Comparing low and high selective relations, notice that for uniform-Gaussian distributions there are a slightly difference in response time, visited and in particular for compatible features. Surprisingly, in this case the response time, the percentage of visited features and compatible features of ITHR $(v_{no\_dj})\_\theta$ for *in* are higher than those of *overlap*. This is due to the fact that for threshold v_{no_dj}, *in* considers in the relaxing process more relations than *overlap*.

5.4.2.2 Impact of the MBR Size of Dataset

As presented for skewed distribution, for this experiments the considered techniques are IBF_θ and ITHR$(v_3)_\theta$ (ITHR(0.8)_θ in our case). The aim is to analyze the impact of the MBR size of datasets for both a high selective (*in*) and a low selective (*overlap*) relations.

Considering Figures 5.28, the same considerations pointed out for the case uniform-skewed distributions holds. Thus, the response time, the percentage of nodes, visited features



(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

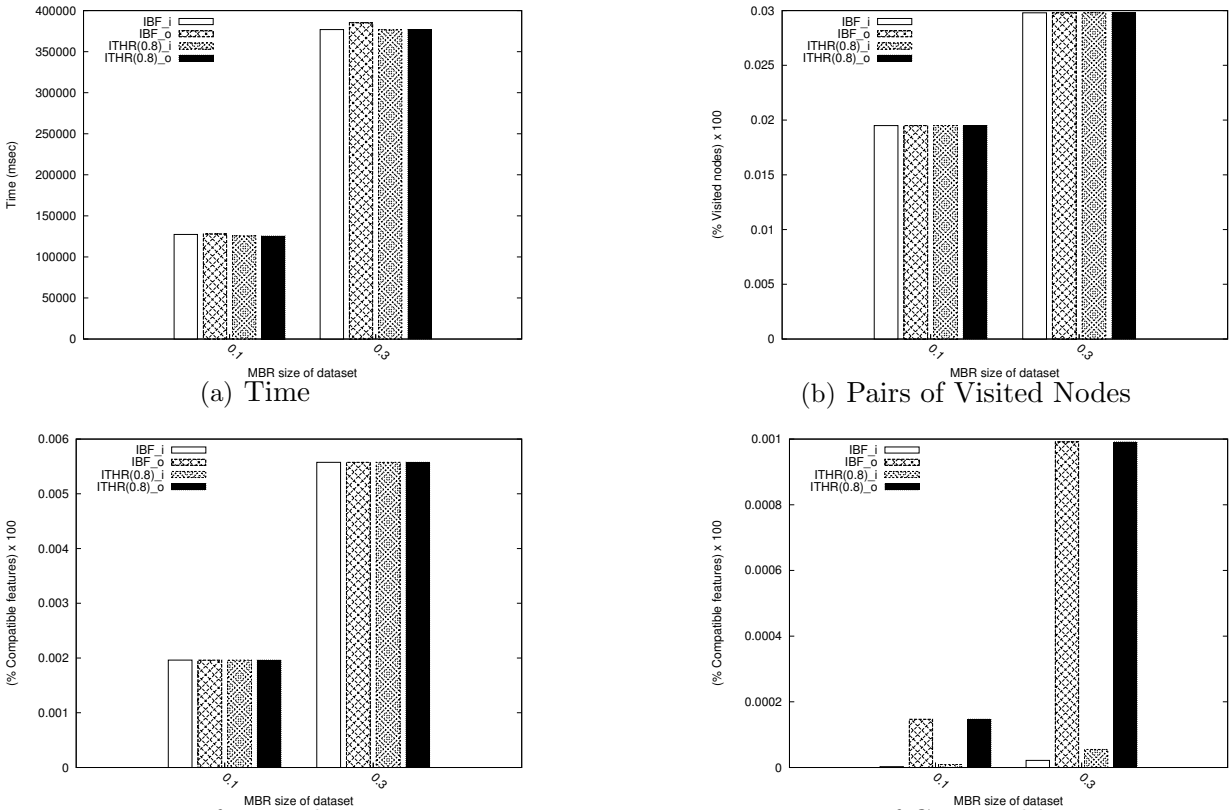
Figure 5.27: Relaxed methods with respect to non relaxed ones over uniform-Gaussian datasets MBR size ≤ 30, relation *overlap* (low selective relation).

and visited nodes grow up while increasing the MBR size, since MBR density is higher. Moreover, for the low selective relation *overlp* all the techniques have higher response time, percentage of visited nodes, visited features and compatible features w.r.t. those obtained of the high selective relation.

Moreover, it is possible to notice again that the overall statistics for this case are higher than the statistics for the case uniform-skewed, due to the different characteristics of the two distributions.

5.4.2.3 Impact of Threshold Value

The aim of these experiments is to evaluate the impact of the threshold value on the technique ITHR_θ. We consider the response time in msec of ITHR(ρ)_θ by considering four threshold values ($v_{no\_relax}, v_{P2}, v_3, v_{no\_dj}$), considering only the MBR dataset dimension 30. Also in this case, the relation *in* is considered for its characteristics already discussed



(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

Figure 5.28: Impact of the datasets MBR size for uniform-Gaussian distributions.

for the selection query.

As expected, also in this case by increasing the threshold value performance improves since less relations are taken into account during the search, as highlighted in Table 5.10.

Response time in msec				
Dataset size	**0.5**	**0.6**	**0.8**	**1**
10000	27487	27003	26590	26356
20000	82188	79248	78125	77602
30000	168644	163473	160712	160329
40000	262018	253181	248875	247986
50000	400064	383247	376656	374628

Table 5.10: Impact of threshold value for uniform-Gaussian distributions.

5.4.3 Case Uniform-Uniform

The aim of this group of experiment is to compare IBF_θ, ITHR(v_3)_θ (ITHR(0.8)_θ for the considered similarity function) while changing the overlap rate between the two considered datasets. The presented results refer to *overlap* relation since it is the relation most influenced by the variation of the overlap area of the joined datasets.

As expected (see Figure 5.29) for both dimensions (12 and 30) decreasing the rate of overlap between the two datasets the response time, the percentage of visited nodes, visited features and compatible features significantly decrease for both IBF_θ and ITHR(v_3)_θ. As already discussed we have chosen *overlap* since it is the relation most influenced by the variation of the overlap area but, on the other side since it is a low selective relation, there is no difference between ITHR_θ (independently from the threshold value used) and IBF_θ, thus their lines in the graphs have the same trend.

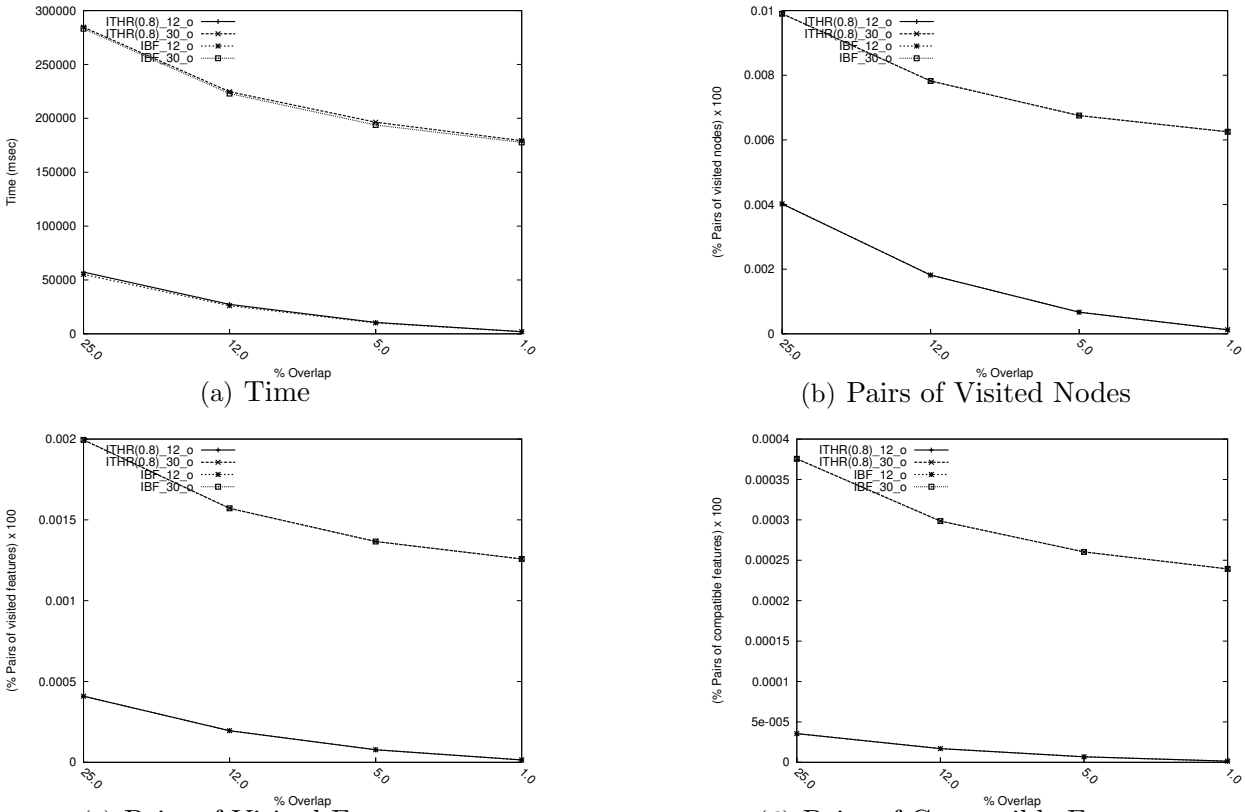
By comparing results obtained with MBR of maximum size 12 and maximum size 30, we notice that for size 12 the object distribution is less dense than for dimension 30. Thus, as expected, the response time, the percentage of visited nodes, visited features and compatible features for dimension 12 is significantly lower than for dimension 30.

5.4.4 Case Skewed-Gaussian

The aim of these experiments is perform all the following techniques, IBF_θ, ITHR(v_{no_relax})_θ, ITHR(v_{no_dj})_θ in order to analyze their behaviours at the variations of the number of pairs considered in the join operation.

To this purpose we consider the predicate *overlap* that is the most similar to the the relation *intersection* usually considered in the computation of spatial joins. Moreover in order to vary the number of considered pairs we apply two approaches: one is to vary the density of the joined datasets, thus we consider two different MBRs sizes: 12 and 30 units; the other approach we consider is the variation of the overlap area between the two datasets, joining datasets with different distributions. Thus, we consider the following cases: (i) uniform-uniform, with an overlap area of at most the 12% of the global area (denoted with IUU); (ii) uniform-Gaussian, with an overlap area of the 100% (denoted with IUG); (iii) uniform-skewed, with an overlap area of the 100% (denoted with IUS); (iv) skewed-Gaussian, with an overlap area of the 100% (denoted with ISG).

As expected, for both sizes 30 and 12 increasing the number of intersections, that is increasing the dataset size, also the response time grows up. On the other side, the percentage of nodes, compatible and accessed features do not have significant variation increasing the dataset size since it represent a rate of the total.



(a) Time (b) Pairs of Visited Nodes

(c) Pairs of Visited Features (d) Pairs of Compatible Features

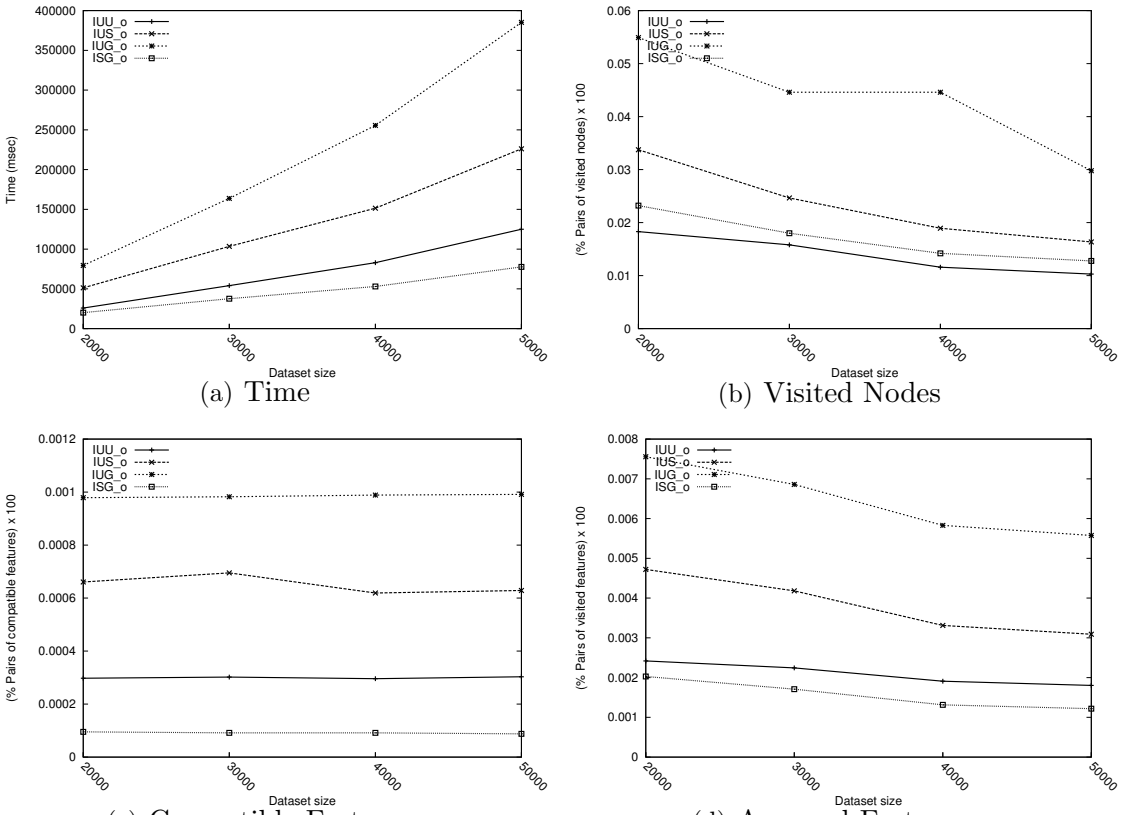
Figure 5.29: Impact of overlap rate for datasets with MBR size ≤ 12 and ≤ 30.

Figures 5.30, 5.31, 5.32 show the results for size 30 respectively for IBF_θ, ITHR(v_{no_relax})_θ, ITHR(v_{no_dj})_θ, while Figures 5.33, 5.34, 5.35 show the results for size 12 respectively for IBF_θ, ITHR(v_{no_relax}), ITHR(v_{no_dj}).

Comparing the different cases of dataset distributions we notice the worst cases are those with maximum overlap between the two datasets, that is uniform-Gaussian and uniform-skewed, while uniform-uniform has only the 12% of overlap and the best case is skewed-Gaussian since the percentage of pairs of visited nodes, visited features and compatible features is the lowest.

5.5 Summary

In this Chapter, we have presented a prototype system called ARTjQA (Advanced Relaxed Topological java Query Analyzer), a configurable application for executing topological relaxed selection and join queries against geo-spatial datasets. ARTjQA relies on a modular



(a) Time (b) Visited Nodes

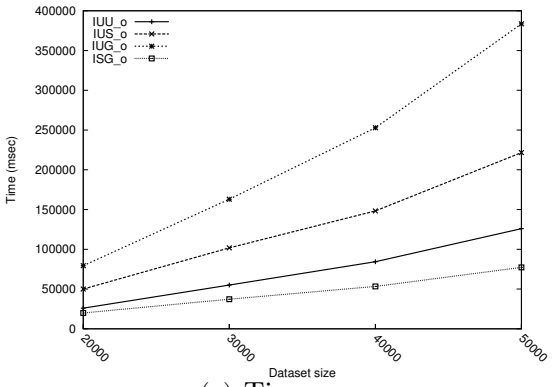
(c) Compatible Features (d) Accessed Features

Figure 5.30: Execution of IBF for *overlap* with datasets MBR size ≤ 30.

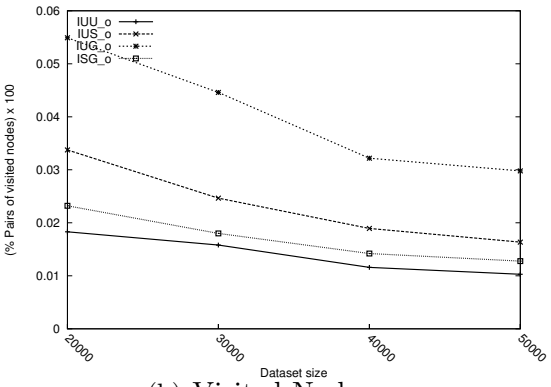
architecture, which makes it easily extensible and modifiable, it implements the proposed index-based algorithms for both selection and join presented in Chapter 4, based on a similarity function specifically defined for relaxation purposes, defined in Section 3.2.3.

In order to deeply test the proposed techniques, we have performed a wide range of experiments using randomly generated datasets, in order to ensures a very high flexibility which cannot be guaranteed by using the real geo-spatial datasets at our disposal. Using adequate synthetic datasets, we created various scenarios, stressing different relevant aspects of relaxed query processing, such as the dimension, the number, and the distribution of geo-spatial data.

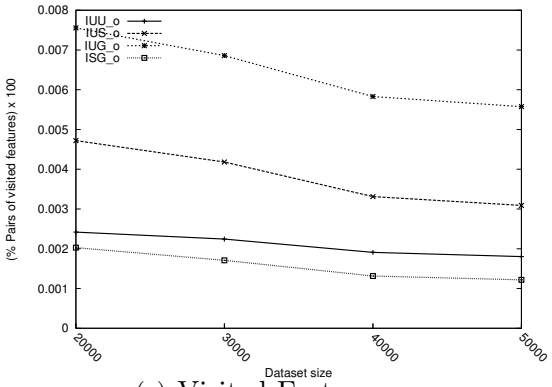
Results show that, as expected, with the exception of relation *disjoint*, the index-based implementation is quite efficient under all the semantics. The difference in performance between relaxed and non relaxed processing is more evident for low selective predicates and is anyway compensated by the improvement in result completeness and accuracy. In all the considered cases, the overhead given by query relaxation is acceptable.



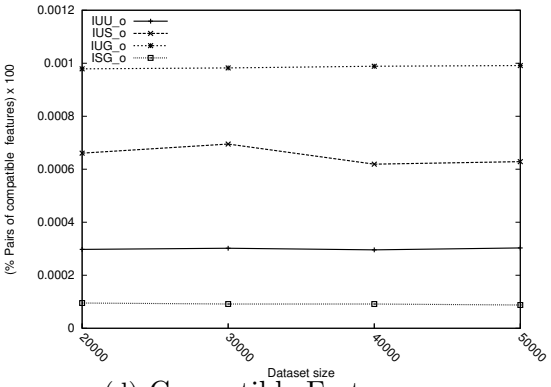
(a) Time



(b) Visited Nodes

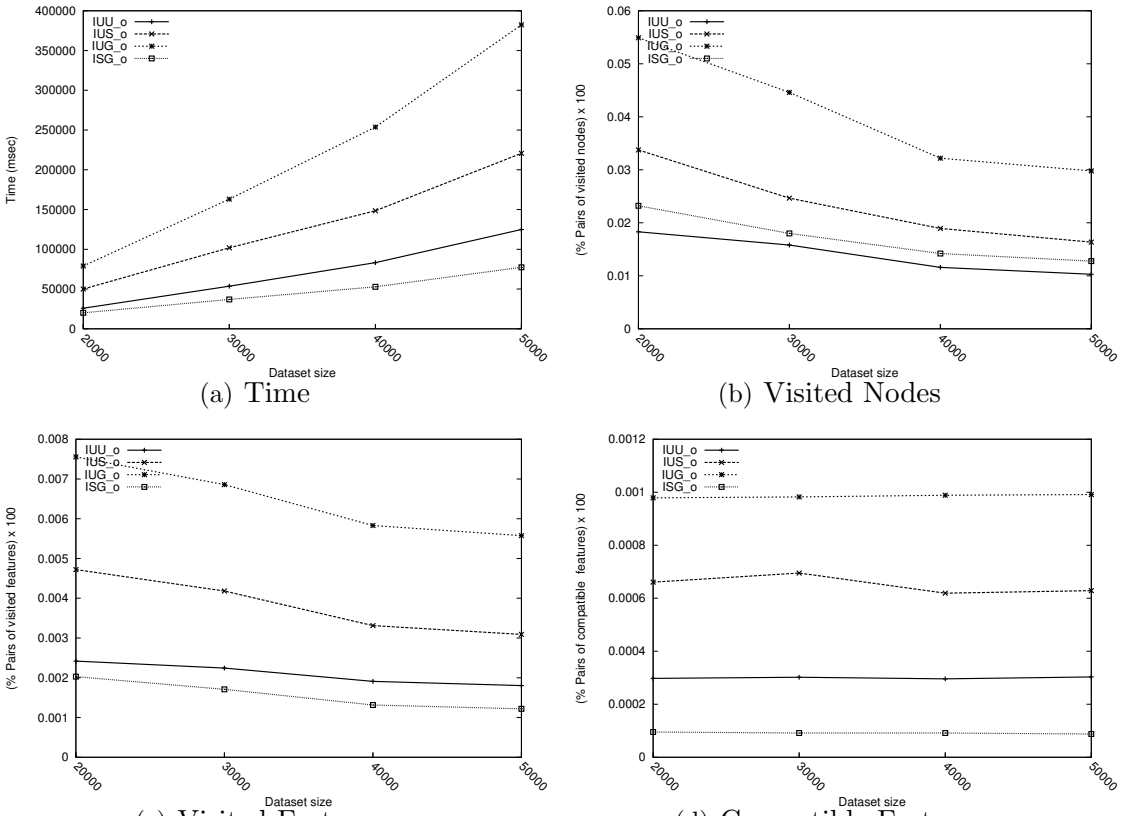


(c) Visited Features



(d) Compatible Features

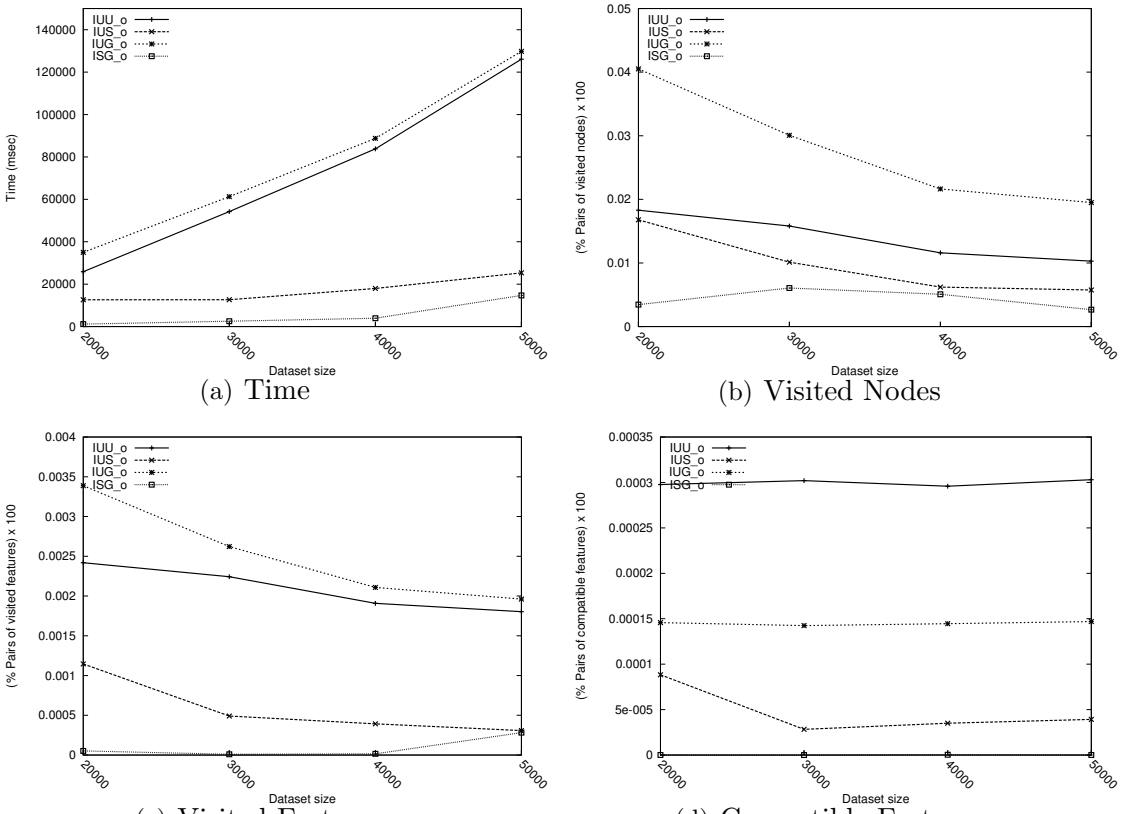
Figure 5.31: Execution of ITHR(1) for *overlap* with datasets MBR size ≤ 30.



(a) Time (b) Visited Nodes

(c) Visited Features (d) Compatible Features

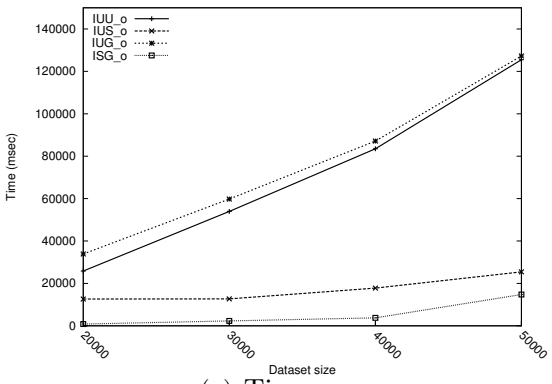
Figure 5.32: Execution of ITHR(0.5) semantics for *overlap* with datasets MBR size ≤ 30.



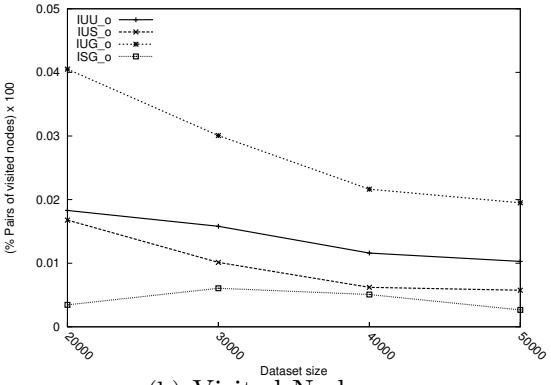
(a) Time (b) Visited Nodes

(c) Visited Features (d) Compatible Features

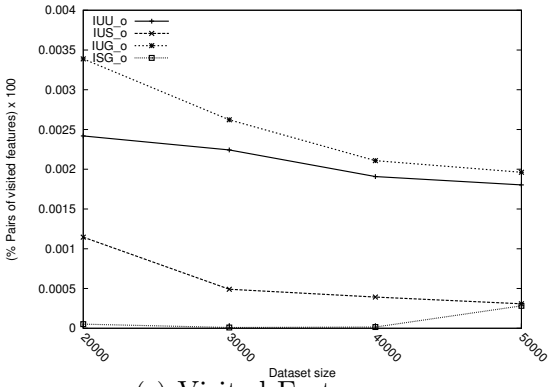
Figure 5.33: Execution of IBF for *overlap* for datasets with MBR size $\leq$f 12.



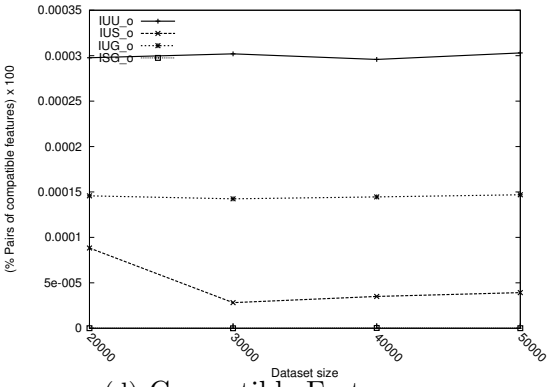
(a) Time



(b) Visited Nodes

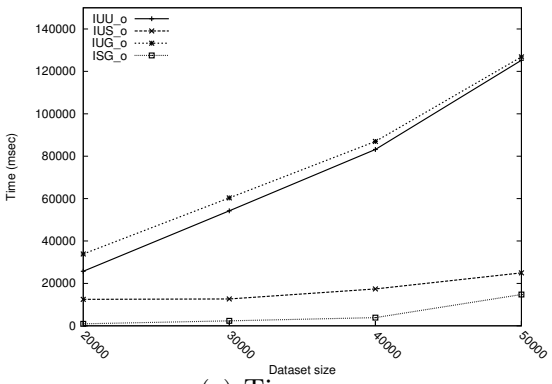


(c) Visited Features

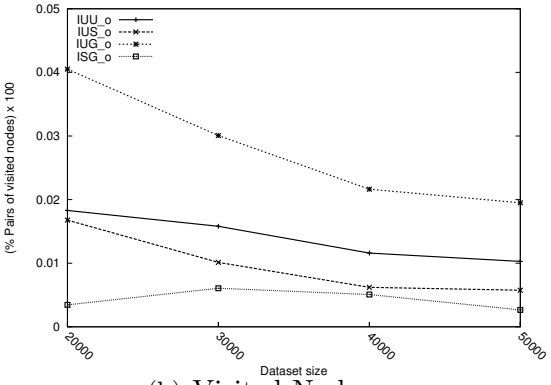


(d) Compatible Features

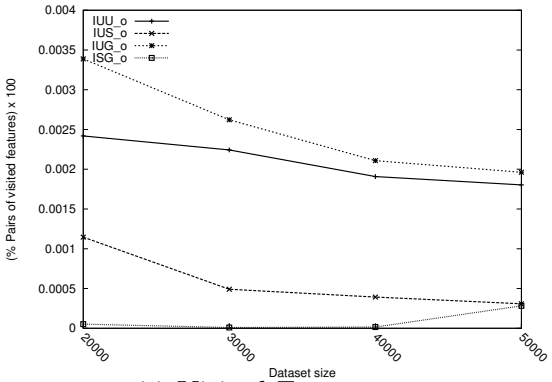
Figure 5.34: Execution of ITHR(1) for *overlap* datasets with MBR size ≤ 12.



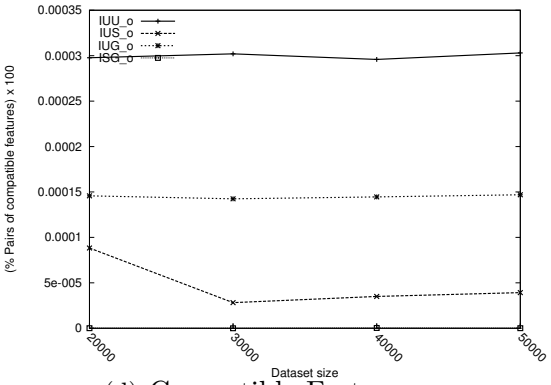
(a) Time



(b) Visited Nodes



(c) Visited Features



(d) Compatible Features

Figure 5.35: Execution of ITHR(0.5) for *overlap* datasets with MBR size ≤ 12.

Chapter 6

Conclusions

In this thesis, we have presented a general framework, relying on spatial relations, for map comparison and for relaxing topological query operators in presence of multi-resolution data. We conclude this thesis by briefly reviewing the main contributions of the PhD research work (Section 6.1) and by discussing the open issues for further research activities (Section 6.2).

6.1 Summary of the Contributions

The major contributions of this thesis can be summarized as follows.

- An extension of the existing models for topological and cardinal relations has been proposed, by considering objects with arbitrary type (points, lines, and regions), in order to cope with a larger number of configurations, which often arise in advanced applications of geo-spatial data.
- Based on the proposed models, a general framework for the comparison, in terms of consistency and similarity, of multi-resolution geo-spatial datasets has been introduced. The framework relies on the usage of specific similarity functions, defined in the context of the thesis work, taking into account object dimensions and their mutual spatial (i.e., topological and cardinal) relations.

  For topological relations, a discussion concerning why the proposed function is not suitable to be used in relaxed query processing has also been provided. Starting from the identified drawbacks, an additional similarity function has been presented, to be used for query relaxation.

- A general strategy for relaxing geo-spatial selection and join operators has also been proposed, providing three distinct semantics, each addressing a different relaxation issue. Query processing algorithms have then been proposed for all the considered operators and semantics. They rely on the usage of the R-tree index structure and are based on a branch and bound approach, which discards the visit of some R-tree sub-trees that cannot produce further results.
- In order to show the efficacy and efficiency of the proposed relaxed query operators, a prototype system, called ARTjQA (Advanced Relaxed Topological java Query Analizer), has finally been developed. A complete set of experimental results have also been presented and discussed, showing that the overhead given by query relaxation is acceptable.

Some of the results achieved in the context of this PhD thesis have already been presented - or will be presented in the next future - at the international scientific community. In particular, the framework for multi-resolution maps consistency checking has been presented in [BCP05], for topological relations, and in [PCB07], for both topological and cardinal relations.

Concerning relaxed geo-spatial query processing techniques, in [BBC+06] an earlier definition of the Threshold and Nearest Neighbor semantics is presented, through the definition of the so-called *weak* and *Nearest Neighbor* predicates.

6.2 Topics for Further Research

Multi-resolution maps comparison and relaxed geo-spatial query processing are two very important issues for advanced geo-spatial applications. Besides the approaches proposed in this thesis, additional issues have to be taken into account in order to cope with such problems in an effective and complete way. In particular, two different directions for future work can be devised: the first one concerns the extension of the proposed prototype ARTjQA; the second one concerns additional issues aimed at consolidating the proposed frameworks. Both research directions are briefly discussed in the following.

6.2.1 ARTjQA extensions

In order to make ARTjQA an effectively usable system for relaxed query processing over multi-resolution geo-spatial maps, some functionalities have to be consolidated and expanded. In particular:

1. A detailed experimentation of ARTiQA, customized using various similarity functions, over real geo-spatial datasets is needed, in order to assess results obtained for synthetic datasets on real geo-spatial applications.

2. The integration of the relaxation query processing algorithms, implemented in ARTjQA, inside open source geo-spatial database management systems, such as PostGIS [pos], and in geo-spatial mediator systems, such as VirGIS [BEL+04], is another issue that we claim should be investigated in order to make relaxed geo-spatial query processing a practical technology.

3. Several ARTjQA functionalities could be extended, in order to make the prototype more user-friendly. Among them, we recall the development of a graphical user interface, for easily analyzing the impact of relaxation on query results. The development of a new ARTjQA version as Web service, relying on OGC standards, is another issue that we plan to take into account in the future.

6.2.2 Open Issues

In the following, further topics for future research work on management of multi-resolution geo-spatial data based on spatial relations are briefly discussed.

Similarity functions. The consistency and similarity framework and the proposed relaxed query processing techniques rely on the usage of some similarity functions for spatial relations. In this thesis, we have presented some of these functions, discussing in which context they can be more suitably used. The usage of such functions should however be validated with respect to user needs and expectations. Additionally, in order to evaluate the generality and the flexibility of the proposed comparison and query processing techniques, other sets of topological and cardinal relations as well as other similarity functions should be taken into account during the analysis of the proposed approaches. Finally, the definition of a similarity function for cardinal direction relations, to be used for relaxed query processing, is an additional topic left to future work.

Complex conditions and skyline semantics. The relaxed operators considered in this thesis rely on single topological conditions. An interesting extension concerns the definition of query processing techniques for relaxed geo-spatial selection and join operators based on conjunctions of conditions. We claim that, in this case, operators semantics could be defined taking into account the concept of *skyline* [KLTV06, BKS01a]. The skyline semantics, similarly to the Best Fit semantics for simple conditions, would return the objects which better satisfy *all* the conditions in the query.

Spatial knowledge discovery. Spatial knowledge discovery can be defined as the process of discovery knowledge from spatial databases. Unfortunately, the application of traditional data mining techniques to spatial data mining is not straightforward. This is due to several factors such as the complexity of spatial data types, the presence of both spatial and non-spatial attributes, the existence of spatial auto-correlations (that is, the trend of spatial data to be highly self correlated).

Even if several approaches for spatial data mining have already been proposed, a lot of papers ([Mil04, RHE+04, KH96]) claim that there are still several challenges concerning spatial (geographic) knowledge discovery that should be addressed. We believe that the proposed consistency and similarity framework can be used in this context for the extraction of interesting and previously unknown knowledge from multi-resolution spatial datasets.

Stream-based progressive processing. Several innovative applications require the management of data which is not totally stored but available to the application as a continuous stream. This is for example the case of data generated by sensors and managed inside sensor networks. Also geo-spatial data could be delivered as a stream. As example is given by geo-spatial positions related to car accidents, acquired in real-time. Query processing over data streams is approximated by definition. As a consequence, we believe that the investigation of how the proposed relaxed operators can be processed in a continuous way over geo-spatial data streams is another issue we plan to investigate in the future.

Spatial data quality and user satisfaction. Data quality of query results, in terms of accuracy, completeness and satisfaction of user needs and expectations, are very critical issues for applications. In particular, in geo-spatial context data quality is important when relaxing geo-spatial queries. The investigation of the quality of data, obtained as result of relaxed topological operator execution, is another issue we plan to address.

Bibliography

[Ale61] P. Alexandroff. *Elementary Concepts of Topology Book Description.* Dover Publications, 1961.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communication ACM*, 26(11):832–843, 1983.

[arc] ArcGIS Working With Geodatabase Topology. http://www.esri.com/library/whitepapers/pdfs-/geodatabasetopology.pdf.

[AS91] W. G. Aref and H. Samet. Extending a DBMS with Spatial Operations. In *SSD'91: Proc. of the Second International Symposium on Advances in Spatial Databases*, pages 299–318, London, UK, 1991. Springer-Verlag.

[BBB+04] A Belussi, E. Bertino, Catania B., M. L. Damiani, and Nucita A. An authorization model for geographical maps. In *ACMGIS'04: Proc. of the 12th ACM International Symposium on Geographic Information Systems*, pages 82–91, 2004.

[BBC+06] A. Belussi, O. Boucelma, B. Catania, Y. Lassoued, and P. Podestà. Towards Similarity-Based Topological Query Languages. In *LNCS 4254: Proc. of EDBT Workshops*, pages 675–686, 2006.

[BCG02] N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. Database Syst.*, 27(2):153–187, 2002.

[BCP05] A. Belussi, B. Catania, and P. Podestà. Towards topological consistency and similarity of multiresolution geographical maps. In *ACMGIS'05: Proc. 13th ACM International Work. on Geographic Information Systems*, pages 220–229, 2005.

[BE96] H. T. Burns and M. J. Egenhofer. Similarity of Spatial Scenes. In *Proc. of the 7th International Symposium on Spatial Data Handling*, pages 31–42, 1996.

[BE01] M. Bertolotto and M. J. Egenhofer. Progressive Transmission of Vector Map Data over the World Wide Web. *Geoinformatica*, 5(4):345–373, 2001.

[BEL+04] O. Boucelma, M. Essid, Z. Lacroix, J. Vinel, J-Y. Garinet, and A. Betari. VirGIS: Mediation for Geographical Information Systems. In *Proc. of the IEEE International Conference on Data Engineering*, pages 855–856, 2004.

[Ber98] M. Bertolotto. *Geometric Modeling of Spatial Entities at Multiple Levels of Resolution.* PhD thesis, Departement of Informatic Science and Informatic, University of Genoa, Italy, 1998.

[BKS93] T. Brinkhoff, H. P. Kriegel, and B. Seeger. Efficient processing of spatial joins using R-trees. In *SIGMOD'93: Proc. of the 1993 ACM SIGMOD International Conference on Management of data*, pages 237–246, New York, NY, USA, 1993. ACM.

[BKS01a] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Proc. of the 17th International Conference on Data Engineering*, pages 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[BKS01b] S. Brzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proc of the International Conference on Data Engineering*, pages 421–430, 2001.

[BKSS90] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD'90: Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 322–331, 1990.

[CDF96] E. Clementini and P. Di Felice. A Model for Representing Topological Relationships Between Complex Geometric Features in Spatial Databases. *Information Sciences*, 90(1-4):121–136, 1996.

[CDFVO93] E. Clementini, P. Di Felice, and P. Van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proc. of the 3rd International Symposium on Advances in Spatial Databases (SSD)*, pages 277–295, 1993.

[CG99] S. Chaudhuri and L. Gravano. Evaluating top-k selection queries. In *VLDB'99: Proc. of the 25th International Conference on Very Large Data Bases*, pages 397–410, San Francisco, CA, USA, 1999.

[CMTV00] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Closest pair queries in spatial databases. In *SIGMOD Conference*, pages 189–200, 2000.

[CMTV04] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. Algorithms for processing k-closest-pair queries in spatial databases. *Data Knowl. Eng.*, 49(1):67–104, 2004.

[DMP93] L. DeFloriani, P. Marzano, and E. Puppo. Spatial Queries and Data Models. In *Proc. of Conference On Spatial Information Theory*, pages 113–138, 1993.

[EAT92] M. J. Egenhofer and K. Al-Taha. Reasoning about Gradual Changes of Topological Relationships. In *LNCS 639: Proc. of Theory and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 196–219, 1992.

[ECDF94] M. J. Egenhofer, E. Clementini, and P. Di Felice. Evaluating Inconsistency Among Multiple Representations. In *Proc. of the 6th International Symposium on Spatial Data Handling*, pages 143–160, 1994.

[EF91] M. J. Egenhofer and R. D. Franzosa. Point-set Topological Spatial Relations. *International Journal of Geographic Information Systems*, 5(2):161–174, 1991.

[EF95] M. J. Egenhofer and R. D. Franzosa. On the Equivalence of Topological Relations. *International Journal of Geographic Information Systems*, 9(2):133–152, 1995.

[EFJ90] M. J. Egenhofer, A.U. Frank, and J. P. Jackson. A topological data model for spatial databases. In *SSD'90: Proc. of the first symposium on Design and implementation of large spatial databases*, pages 271–286, 1990.

[Ege89] M. J. Egenhofer. A formal definition of binary topological relationships. In *FODO 1989: Proc. of the 3rd International Conference, on Foundations of Data Organization and Algorithms*, pages 457–472, 1989.

[EGG+99] M. J. Egenhofer, J. Glasgow, O. Gunther, J. R. Herring, and D. Peuquet. Progress in computational methods for representing geographical concepts. *International Journal of Geographical Information Science*, 13(8):775–796, 1999.

[EH90] M. J. Egenhofer and J. Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical Report Univ.of Maine, Orono, TECHREP: 90-12, 1990.

[EH91] M. J. Egenhofer and J. Herring. *Geographical Information Systems, Vol. 1: Principles*, chapter High-Level Spatial Data Structures, pages 227–237. Longman, 1991.

[EM95] M. J. Egenhofer and D. Mark. Modeling Conceptual Neighborhoods of Topological Line-Region Relations. *International Journal of Geographic Information Systems*, 9(5):555–565, 1995.

[FK86] A.U. Frank and W. Kuhn. Cell graphs: A provable correct method for the storage of geometry. In *Proc. of the Second International Symposium on Spatial Data Handling*, 1986.

[Fra92a] A. U. Frank. Qualitative Spatial Reasoning about Distances and Directions in Geographic Space. *Journal of Visual Languages and Computing*, 3:343–371, 1992.

[Fra92b] A. U. Frank. Spatial Concepts, Geometric Data Models and Geometric Data Structures. *Computer Geosciences*, 18(4):409–417, 1992.

[Fra96] A. U. Frank. Qualitative Spatial Reasoning: Cardinal Directions as an Example. *International Journal of Geographical Information Science*, 10(3):269–290, 1996.

[Fre91] C. Freksa. Conceptual Neighborhood and its role in temporal and spatial reasoning. In *Proc. of the IMACS Work. on Decision Support System and Qualitative Reasoning*, pages 181–187, 1991.

[Gü88] R. H. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In *EDBT'88: Proc. of the International Conference on Extending Database Technology*, pages 506–527, 1988.

[Gü94] R. H. Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399, 1994.

[GB90] O. Guenther and A. Buchmann. Research issues in spatial databases. *SIGMOD Record*, 19(4):61–68, 1990.

[GE00] R. K. Goyal and M. J. Egenhofer. Consistent Queries over Cardinal Directions across Different Levels of Detail. In *Proc. of 11th International Conference on Database and Expert System Applications*, pages 876–880, 2000.

[GE01] R. K. Goyal and M. J. Egenhofer. Similarity of cardinal directions. In *SSTD'01: Proc. of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pages 36–58, London, UK, 2001. Springer-Verlag.

[GG98] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[Goo92] M. F. Goodchild. Geographical data modeling. *Computer Geoscience*, 18(4):401–408, 1992.

[Goy00] R. K. Goyal. *Similarity Assessment for Cardinal Directions Between Extended Spatial Objects*. PhD thesis, Department of Spatial Information Science and Engineering, University of Maine, USA, 2000.

[GS95] R. H. Güting and M. Schneider. Realm-based spatial data types: the ROSE algebra. *The VLDB Journal*, 4(2):243–286, 1995.

[Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD'84: Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[GYC07] M. F. Goodchild, M. Yuan, and T. J. Cova. Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, 21(3):239–260, 2007.

[HCDF95] D. Hernández, E. Clementini, and P Di Felice. Qualitative Distances. In *COSIT'95: Proc. of the Conference On Spatial Information Theory (COSIT)*, pages 45–57, 1995.

[HS99] G.R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 24(2), 1999.

[IAE03] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In *VLDB'2003: Proc. of the 29th International Conference on Very large data bases*, pages 754–765. VLDB Endowment, 2003.

[JS07] E. H. Jacox and H. Samet. Spatial join techniques. *ACM Trans. Database Syst.*, 32(1):7, 2007.

[JTS] JTS Topology Suite. http://www.vividsolutions.com/jts/jtshome.htm.

[KH96] J. Koperski, K. and Adhikary and J. Han. Knowledge Discovery in Spatial Databases: Progress and Challenges. In *roc. ACM SIGMOD Work. on Research Issues on Data Mining and Knowledge Discovery*, pages 55–70, 1996.

[KLTV06] Nick Koudas, Chen Li, Anthony K. H. Tung, and Rares Vernica. Relaxing join and selection queries. In *VLDB'06: Proc. of the 32nd International Conference on Very large data bases*, pages 199–210. VLDB Endowment, 2006.

[Mil04] H. J. Miller. *Geographic data mining and knowledge discovery.* Dover Publications, 2004.

[MP94] C. B. Medeiros and F. Pires. Databases for GIS. *SIGMOD Record*, 23(1):107–115, 1994.

[Mun66] J.R. Munkres. *Elementary differential topology.* Princeton University Press, 1966.

[OGCa] Open GIS Consortium. www.opengis.org.

[OGCb] Simple features specification for sql.

[OGCc] Simple Features Specification SQL. www.opengeospatial.org/standards/sfs.

[ora] Oracle Spatial User's Guide and Reference 10g. http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10826.pdf.

[Par95] J. Paredaens. Spatial Databases, The Final Frontier. In *ICDT'95: Proc. of the 5th International Conference on Database Theory*, pages 14–32, 1995.

[PCB07] P. Podestà, B. Catania, and A. Belussi. *Modelling and Management of Geographical Data over Distributed Architectures.*, chapter Using Qualitative Information in Query Processing over Multiresolution Maps., pages 159–186. Springer-Verlag, Berlin Heidelberg, 2007.

[PD95] E. Puppo and G. Dettori. Towards a Formal Model for Multi-Resolution Spatial Maps. In *SSD'95: Proc. of the 4th International Symposium on Advances in Spatial Databases*, pages 152–169, London, UK, 1995. Springer-Verlag.

[Pig94] S. Pigot. Generalized Singular 3-Cell Complexes. In *SDH'94'95: Proc. of International Symposium on Spatial Data Handling*, pages 89–111, 1994.

[pos] PostGIS 1.4.0 Manual. http://postgis.refractions.net/download/postgis1.4.0.pdf.

[PTS94] D. Papadias, Y. Theodoridis, and T. K. Sellis. The retrieval of direction relations using r-trees. In *DEXA'94: Proc. of the 5th International Conference on Database and Expert Systems Applications*, pages 173–182, London, UK, 1994. Springer-Verlag.

[PTSE95] D. Papadias, Y. Theodoridis, T. Sellis, and M. J. Egenhofer. Topological relations in the world of minimum bounding rectangles: a study with R-trees. In *SIGMOD'95: Proc. of the 1995 ACM SIGMOD International Conference on Management of data*, pages 92–103, 1995.

[RHE+04] J. F. Roddick, E. Hoel, M. J. Egenhofer, D. Papadias, and B. Salzberg. Spatial, temporal and spatio-temporal databases - hot issues and directions for phd research. *SIGMOD Rec.*, 33(2):126–131, 2004.

[rtr] R-tree Portal. http://www.rtreeportal.org/.

[SA95] H. Samet and W. G. Aref. Spatial data models and query processing. pages 338–360, 1995.

[Sam95] H. Samet. *Modern Database Systems, The Object Model, Interoperability and Beyond.*, chapter Spatial data structures., pages 361–385. ACM Press and Addison-Wesley, New York, USA, 1995.

[Sam05] H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., 2005.

[SCR+99] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C. Lu. Spatial Databases-Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):45–55, 1999.

[SK02] S. Skiadopoulos and M. Koubarakis. Consistency Checking for Qualitative Spatial Reasoning with Cardinal Direction. In *Proc. of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 341–355, 2002.

[SK04] S. Skiadopoulos and M. Koubarakis. Composing Cardinal Direction Relations. *Artificial Intelligence*, 152(2):537–561, 2004.

[SRF87] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *VLDB'87: Proc. of the International Conference on Very Large Data Bases*, pages 507–518, 1987.

[SS06] M. Sharifzadeh and C. Shahabi. The Spatial Skyline Queries. In *VLDB'06: Proc. of the International Conference on Very Large Data Bases*, pages 751–762, 2006.

[TE96] N. Tryfona and M. J. Egenhofer. Multi-Resolution Spatial Databases: Consistency Among Networks. In *Proc. of the 6th International Work. on Foundations of Models and Languages for Data and Objects*, pages 119–132, 1996.

[TE97] N. Tryfona and M. J. Egenhofer. Consistency among Parts and Aggregates: A Computational Model. *Transactions in GIS*, 1(3):189–206, 1997.

[WD04] M. Worboys and M Duckham. *GIS: A Computing Perspective*. 2004.

[YDMV07] M.L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis. Top-k Spatial Preference Queries. In *ICDE'07: Proc. of the 23rd International Conference on Data Engineering*, pages 1076–1085, 2007.

[ZPZL05] M. Zhu, D. Papadias, J. Zhang, and D. L. Lee. Top-k spatial joins. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):567–579, 2005.

Appendix A

Example of Input and Configuration Files of ARTjQA

```
<COMPATIBLETABLENODESSELECTION>
  <NAME>
  compatibnodesselection.txt
  </NAME>
  <PATH>
  ../../../..
  </PATH>
</COMPATIBLETABLENODESSELECTION>
<COMPATIBLETABLELEAVESSELECTION>
  <NAME>
  compatibleavessselection.txt
  </NAME>
  <PATH>
  ../../../..
  </PATH>
</COMPATIBLETABLELEAVESSELECTION>
<SIMILARITYFUNCTIONS>
  <NAME>
  similarityfunctions.txt
  </NAME>
  <PATH>
  ../../../..
  </PATH>
</SIMILARITYFUNCTIONS>
<COMPATIBLETABLENODESJOIN>
  <NAME>
  compatibleavesjoin.txt
  </NAME>
<PATH>
../../../..
</PATH>
</COMPATIBLETABLENODESJOIN>
<COMPATIBLETABLELEAVESJOIN>
  <NAME>
  compatibleavesjoin.txt
  </NAME>
  <PATH>
  ../../../..
  </PATH>
</COMPATIBLETABLELEAVESJOIN>
```

Figure A.1: ARTjQA XML configuration file

```
    <JOB>
      <QUERY>
        <QUERYTYPE>
          <TYPE>selection/join</TYPE>
        </QUERYTYPE>
        <QUERYDATA>
          <RTREE1>
            <TYPE> medium </TYPE>
            <PATH> ../../..</PATH>
            <NAME>rtree1.dat</NAME>
          </RTREE1>
          <DATASETRTREE1>
            <PATH>../../..</PATH>
            <NAME>dataset1.dat</NAME>
          </DATASETRTREE1>
          <QUERYOBJECT>
            <PATH>../../..</PATH>
            <NAME>queryobject.tx</NAME>
            <DIMENSION>2</DIMENSION>
          </QUERYOBJECT>
          <VISIT>
            <TYPE>index</TYPE>
          </VISIT>
          <SEMANTIC>
            <TYPE>BF</TYPE>
            <THRVALUE></THRVALUE>
          </SEMANTICS>
          <QUERYCONDITION>
            <TYPE>overlap</TYPE>
            <OPERATOR></OPERATOR>
          </QUERYCONDITION>
          <RESULTFILES>
            <PATH>../../..</PATH>
          </RESULTFILES>
      </QUERYDATA>
    </QUERYD>
  </JOB>
```

Figure A.2: ARTjQA XML input file for selection query

```
<JOB>
  <QUERY>
    <QUERYTYPE>
      <TYPE>selection/join</TYPE>
    </QUERYTYPE>
    <QUERYDATA>
      <RTREE1>
        <TYPE>medium</TYPE>
        <PATH>../../..</PATH>
        <NAME>rtree1.dat</NAME>
      </RTREE1>
      <DATASETRTREE1>
        <PATH>../../..</PATH>
        <NAME>dataset1.dat</NAME>
      </DATASETRTREE1>
      <RTREE2>
        <TYPE>small</TYPE>
        <PATH>../../../</PATH>
        <NAME>rtree2.dat</NAME>
      </RTREE2>
      <DATASETRTREE2>
        <TYPE>small</TYPE>
        <PATH>../../..</PATH>
        <NAME>dataset2.dat</NAME>
      </DATASETRTREE2>
      <VISIT>
        <TYPE>index</TYPE>
      </VISIT>
      <SEMANTIC>
        <TYPE>BF</TYPE>
        <THRVALUE></THRVALUE>
      </SEMANTICS>
      <QUERYCONDITION>
        <TYPE>overlap</TYPE>
        <OPERATOR></OPERATOR>
      </QUERYCONDITION>
      <RESULTFILES>
        <PATH>../../..</PATH>
      </RESULTFILES>
  </QUERYDATA>
 </QUERY>
</JOB>
```

Figure A.3: ARTjQA XML input file for join query