Spatial queries and data models*

Leila De Floriani - Paola Marzano
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova
Viale Benedetto XV, 3 - 16132 Genova - ITALY

Enrico Puppo
Istituto per la Matematica Applicata
Consiglio Nazionale delle Ricerche
Via De Marini, 6 (Torre di Francia) - 16149 Genova - ITALY

Abstract

We present a unified framework for classifying and answering spatial queries relevant to a Geographic Information System. We classify spatial queries into topological, set-theoretic, and metric queries, on the basis of the kind of relationships between the query object and entities in the search space involved. For answering such queries, we propose an approach that combines an object-based description of spatial entities, provided by a topological model, with a partition of the space embedding such entities, given by a spatial index. In particular, we propose a new unified topological model, called the Plane Euclidean Graph (PEG), that is capable of describing point, line, and region data, and that incorporates relational operators on such entities. We briefly describe major techniques, rooted in computational geometry, for solving interference queries and overlays on such a data model. Finally, we describe the use of a superimposed spatial index for speeding up searches and answering queries involving distances.

1 Introduction

One main concern for a spatial database supporting a Geographical Information System is the ability of handling spatial data and of integrating them with other kinds of information (e.g., tabular data). This requirement involves some fundamental topics, like the definition and classification of spatial relationships

*This work has been supported by a research grant of the Italian National Research Council.

and queries, the development of efficient data models and structures for representing spatial data, and the design of efficient algorithms for solving geometric problems involved in answering spatial queries.

The above topics correspond to complementary research streams, that often evolved independently. The classification of spatial relations starts from classical topology [30], and has been object of research in the context of GIS during the last years [7, 8, 18, 9, 33]. The development of spatial models has been deeply studied both in geometric modeling (e.g., [25, 21, 28, 27]), in computational geometry (e.g., [13, 24, 14, 5]), and, specifically, in GIS (e.g., [10, 15, 6, 32, 22, 33, 23]). Finally, the efficient solution of geometric problems is mainly investigated in computational geometry (e.g., [24]).

At the state-of-the-art, several geographic information systems have been developed that are able to answer "some" spatial queries on geographic data. Anyway, though some work has been done towards an integration of spatial relationships, data models, and geometric algorithms (e.g., [7, 22, 16, 29, 33]), the concordance between a complete and integrated formalization and an efficient implementation is still an open issue.

The basic aim of this paper is to investigate the formalization of spatial queries, and to propose a combined object-based and space-based repre

sentation of spatial entities, that is able to support suitable algorithmic techniques for the solution of geometric problems involved in answering spatial queries.

Informally, spatial queries can be defined as queries about spatial relations (e.g.: intersection, containment, boundary, adjacency, proximity, etc.) of entities geometrically defined and located in space. Thus, a first problem arising when analyzing queries and operations relevant to a given application, is to define the spatial entities and their possible relationships in the context of the application considered. In Geographic Information Systems, we essentially deal with plane maps composed of three basic spatial entities (points, lines, and regions), which have a well-defined geometric characterization. Here, we propose a formal framework for characterizing spatial entities and relations, and we give a classification of spatial relations within such a framework. The classification we adopt is similar, though not coincident, to classifications proposed by other authors [7, 33], and it is aimed to the identification of three classes that, besides being characterized by the different underlying geometric concepts, reflect the different computational problems met in answering the corresponding queries.

Topological relations encode the connecting structure of a map and can be further classified into adjacency, boundary, and co-boundary relations among the basic spatial entities. Set-theoretic relations are based on concepts like inclusion, intersection, coincidence, element-of, and encode spatial interferences between entities, even from different maps. Metric relations involve the concept of distance and encode spatial proximity.

Once spatial relations have been defined, the definition of queries induced by them is straightforward: a spatial query is based on a spatial entity q, called the *query object*, a set of entities S of a map, called the *query space*, and a spatial relation $\Re$. The answer to such a query must report all entities in S that are in relation $\Re$ with q. The algorithmic technique adopted for answering a spatial query is highly dependent of the relation inducing the query and of the data model used to encode spatial entities.

Under an *object-based* approach, the representation of spatial entities is independent of their position in space. Map data can be represented in a model where each entity is described explicitly and exactly (a point by its location, a line segment by its equation and endpoints, a region by its border). When topological relations between entities are stored as well, we obtain a *topological model* that provides a complete structural representation of a map.

In this paper, we propose a unified data model able to describe point, line and region data, either isolated or topologically related, called a *Plane Euclidean Graph (PEG)*. A PEG provides a combined geometric and topological representation of a map, which is invariant under affine transformations, and it incorporates relational operators on such entities.

We show how topological queries and set-theoretic queries about objects within a single map can be efficiently answered through the relational operators of the PEG. Other interference queries, involving objects from different maps, can be solved through efficient and effective algorithmic techniques rooted in computational geometry. We give a brief description of such techniques, and we also show how they can be used for answering multiple queries and computing map overlays.

The use of a topological model, however, is not sufficient for answering queries involving proximity constraints. Moreover, interference queries and overlays can be sped up by localizing interference computations. This is possible if data are represented in the context of a *space-based* structure. In order to complete our model with one such structure, we superimpose a spatial index on the PEG, which acts as a pruning device in spatial searches.

Spatial indexing structures have been developed by several authors and they are extensively used in GISs [11, 28]. Spatial indices are often superimposed on unstructured sets of spatial entities, without an explicit encoding of their mutual relations [11, 17]. Our proposal is the use of a hybrid model, i.e., a model combining the topological representation provided by the PEG, with a superimposed spatial index, which provides a partition of the space embedding the PEG. A hybrid model supports both an object-based and a space-based approach in representing objects and in answering queries in the context of a GIS.

2 Spatial entities and relations

In this work, we deal with plane *maps*. Entities composing a map are embedded in the plane. Thus, each entity must belong to one of three distinct classes, depending of its dimension:

- 0-dimensional entities form the class **P** of *points*;
- 1-dimensional entities form the class **L** of *lines*;
- 2-dimensional entities form the class **R** of *regions*.

Given a generic class **C**, we denote with dim(**C**) the dimension of entities of **C** (dim(**P**)=0, dim(**L**)=1, and dim(**R**)=2). Elements of each class have a well-defined structural and geometric characterization, that allows a complete and concise description of each entity:

- A *point* is characterized by its coordinates.
- A *line* is characterized by its endpoints, that define its *boundary*, and by its geometry. We have three different types of lines:
  - straight-line segment: there is no need for geometry, since endpoints provide a complete description;
  - polygonal chain: the geometry is represented by a sequence of points in the plane;
  - curve: the geometry is represented by an equation.

  In the following, we will restrict our consideration to the first two types, assuming that curves can be approximated by polygonal chains.
- A *region* is completely characterized by its boundary. The boundary is composed of a set of closed and simple polygonal chains:
  - a chain defines the *outer boundary* of a region;
  - other chains may define *inner boundaries*, separating a region from other regions completely contained into it.

  Notice that a closed chain may either consist of a single line, or be composed of a (closed) sequence of lines; in this latter case, the boundary of a region will contain both such lines and their endpoints. By Jordan theorem, a region can be defined as the portion of plane inside the outer boundary and outside all inner boundaries.

The concept of boundary, defined above, will be used in the following, together with its symmetric concept of co-boundary: the *co-boundary* of an entity O is the set of all entities that contain O in their boundary. Given an entity O, we denote by ∂O the boundary of O, and by γO the co-boundary of O. Notice that the boundary of an object is composed of entities of lower dimension, while its co-boundary is composed of entities of higher dimension. Thus, points have an empty boundary, and regions have an empty co-boundary (in 2D). We also define the *immediate boundary* and *co-boundary*, denoted by $\partial_1 O$ and by $\gamma_1 O$, respectively, as the subsets of ∂O and γO that contain only entities of one dimension lower or one dimension higher than O, respectively. The immediate boundary of regions and the immediate co-boundary of points are composed only of lines.

For now, we define a *map* as a triple $M = (P, L, R)$, where $P \subset \mathbf{P}$, $L \subset \mathbf{L}$, and $R \subset \mathbf{R}$. Notice that, for the consistency of the map, L shall contain at least all lines defining the boundary of entities in R, and, similarly, P shall contain at least all points defining the boundary of entities in L. Notice that a map M does not necessarily contain *only* points and lines that are part of the boundary of some other lines and regions of the map, respectively: isolated points and lines can be present, that denote *features* of the map. For instance, if M is the map of a countryside, relevant regions could denote fields, woods, pastures, etc.; on the other hand, some lines will denote roads and streams, and some points will denote houses, huts, springs, etc., that do not necessarily coincide with points and lines on the border of regions.

In the context of a single map, we do not allow general intersection between spatial entities: we break lines into chains by inserting points in the map wherever a line intersects another line. Such intersection points are introduced not only for convenience, but they usually have some semantics associated with: where a river intersects a road there will be a bridge; where a road intersects the border of a farm there will be a gate.

A special case of a map is the *network*: in a network only lines representing communication channels (e.g., rivers, roads, railways, pipes), and only branches and crossing points of such lines are important, while regions are irrelevant. Thus, a network can be regarded as a map $N = (P, L, \emptyset)$.

In the following Subsections, we will try to formalize and classify relationships between spatial entities that can be relevant to the definition of spatial queries. We propose a classification into three major groups, according to a topological, set-theoretic, or metric point of view. Of course, our classification is not exhaustive (e.g., we do not consider here order relations [7]). Nevertheless, we introduce a framework that can help to formalize other relations on spatial entities, and to define queries based on such relations.

A *relation* $\Re$ is defined between two classes $\mathbf{C}_1$ and $\mathbf{C}_2$; in general, we will denote by $\stackrel{\Re}{\sim}$ an unordered relation, and by $\stackrel{\Re}{\leadsto}$ an ordered relation. Thus, given an ordered relation $\Re$ on $\mathbf{C}_1 \times \mathbf{C}_2$, and two entities $a \in \mathbf{C}_1$ and $b \in \mathbf{C}_2$, we will denote with $a \stackrel{\Re}{\rightsquigarrow} b$ the predicate "a is in relation $\Re$ with b".

2.1 Topological relations

The definition of a spatial entity introduced above induces topological relations between entities of the three classes, namely, relations between an entity and lower dimensional entities of its boundary, or, symmetrically, between an entity and higher dimensional entities of its co-boundary, and relations between entities of the same class that are adjacent through some entity of a different class. Nine *topological relations* can be defined by combining $\mathbf{P}$, $\mathbf{L}$ and $\mathbf{R}$ in pairs. As topological relations are defined through concepts like boundary, co-boundary, and adjacency, we actually consider them in the context of a single map $M = (P, L, R)$, that is consistent with the above concepts[1].

Topological relations can be classified into three different groups, depending of the relative dimensions of the classes involved; each group is composed in turn of three relations.

- *Adjacency relations* are characterized by $\dim(C_1) = \dim(C_2)$.

  PP: let $a, b \in P$, then $a \stackrel{PP}{\sim} b$ iff $\gamma_1 a \cap \gamma_1 b \neq \emptyset$ (i.e., iff there exist a line $l \in L$ such that a and b are endpoints of l);

  LL: let $a, b \in L$, then $a \stackrel{LL}{\sim} b$ iff $\partial a \cap \partial b \neq \emptyset$ (i.e., a and b have an endpoint in common);

  RR: let $a, b \in L$, then $a \stackrel{RR}{\sim} b$ iff $\partial_1 a \cap \partial_1 b \neq \emptyset$ (i.e., there exist a line $l \in L$ separating a from b).

- *Boundary relations* are characterized by $\dim(C_1) > \dim(C_2)$. The following definition holds for the three relations LP, RP, and RL, by proper instantiation of sets C_1 and C_2:

C_1C_2: let $a \in C_1$ and $b \in C_2$, then $a \stackrel{C_1C_2}{\rightsquigarrow} b$ iff $b \in \partial a$.

- *Co-boundary relations* are characterized by $\dim(C_1) < \dim(C_2)$, and they are symmetric with respect to the previous ones. The following definition holds for the three relations PL, PR, and LR, by proper instantiation of sets C_1 and C_2:

C_1C_2: let $a \in C_1$ and $b \in C_2$, then $a \stackrel{C_1C_2}{\rightsquigarrow} b$ iff $b \in \gamma a$.

[1]In the following, we use interchangeably a set C and its corresponding class $\mathbf{C}$, whenever no ambiguity arises.

In Section 4, we will introduce a model and a data structure that allow an efficient encoding and retrieval of information on topological relations involving a given entity of a map. Such relations will be useful in order to answer queries that involve computing the boundary or the co-boundary of objects, or queries that involve navigation of the map by following adjacencies.

2.2 Set-theoretic relations

So far, we have considered spatial entities as topological objects. Under a different point of view, points can be considered as basic elements, and lines and regions can be regarded as infinite and continuous sets of points. This approach allows us to define *set-theoretic relations*, that are based on concepts like coincidence, element-of, inclusion, and intersection, denoted with the usual symbols $\equiv$, $\in$, $\subset$, and $\cap$, respectively. Notice that, in order to compare spatial objects through set operators, they do not need to be part of a given map. Thus, the following relations will be defined in general on the three spatial classes rather than on subsets of such classes composing a map.

We consider four groups of relations, one for each set operator, and, for each group, we consider all possible ordered pairs of classes $\mathbf{C}_1\mathbf{C}_2$ that lead to a meaningful relation. The following definitions are just a straightforward translation of set operators into our formalism.

- *Coincidence relations* are unordered and require $\mathbf{C}_1 \equiv \mathbf{C}_2 \equiv \mathbf{C}$; thus, they can be defined on pairs **PP**, **LL**, and **RR**. A relation is trivially defined as follows:

  $\equiv_{\mathbf{C}}$: let $a, b \in \mathbf{C}$, then $a \overset{\equiv_{\mathbf{C}}}{\rightsquigarrow} b$ iff $a \equiv b$.

- *Element-of relations* are ordered and can be defined on pairs **PL** and **PR**, and, symmetrically, for pairs **LP** and **RP**. The following general definitions hold both for $\mathbf{C} \equiv \mathbf{L}$ and for $\mathbf{C} \equiv \mathbf{R}$:

  $\in_{\mathbf{C}}$: let $a \in \mathbf{P}$ and $b \in \mathbf{C}$, then $a \overset{\in_{\mathbf{C}}}{\rightsquigarrow} b$ iff $a \in b$.

  $\ni_{\mathbf{C}}$: let $a \in \mathbf{C}$ and $b \in \mathbf{P}$, then $a \overset{\ni_{\mathbf{C}}}{\rightsquigarrow} b$ iff $b \in a$.

  Notice that relation $\in_{\mathbf{R}}$ is fundamental for defining of one of the most important and common spatial queries, known as point location.

- *Containment relations* are ordered and are defined for pair **LR** and, symmetrically, for pair **RL** as follows:

  $\subset$: let $a \in \mathbf{L}$ and $b \in \mathbf{R}$, then $a \overset{\subset}{\rightsquigarrow} b$ iff $a \subset b$.

  $\supset$: let $a \in \mathbf{R}$ and $b \in \mathbf{L}$, then $a \overset{\supset}{\rightsquigarrow} b$ iff $b \subset a$.

- *Intersection relations* are defined on all pairs obtained by combining classes **L** and **R** and have the following general definition:

$$\cap_{\mathbf{C}_1\mathbf{C}_2}\text{: let } a \in \mathbf{C}_1 \text{ and } b \in \mathbf{C}_2, \text{ then } a \overset{\cap_{\mathbf{C}_1\mathbf{C}_2}}{\sim} b \text{ iff } a \cap b \neq \emptyset.$$

Notice that, as for topological relations, each possible ordered pair of classes appear in at least one set-theoretic relation. Moreover, pairs **LL** and **RR** appear in both coincidence and intersection relations, and pairs **RL** and **LR** appear in both containment and intersection relations. This is obvious, since coincidence and containment are special cases of set intersection.

2.3 Metric relations

Some important spatial queries, like proximity queries, involve distances between spatial entities. Given a metric d on the plane (usually the Euclidean metric), the distance $d(p, q)$ between two points p and q is extended to other entities X and Y (either lines or regions) as follows:

$$d(p, X) = \min_{q \in X} d(p, q);$$

$$d(X, Y) = \min_{p \in X} d(p, Y).$$

Thus, the distance between a pair of entities is always defined, independently of their classes. Moreover, given an entity O and a group of entities G, if we consider the distance between O and each entity $P \in G$, we can define metric properties of elements in G with respect to O, like having minimum or maximum distance from O, or lying within a given range from O.

In the following, we define some relations based on metric concepts, that are relevant to the definition of spatial queries. Notice that metric relations are not absolute relations between a pair of entities, but they are relative relations defined with respect to either a reference set (min/max relations), or to a reference value (range relations).

- *Min/max relations* are ordered relations defined between a pair of entities $a \in \mathbf{C}_1$ and $b \in \mathbf{C}_2$, and with respect to a reference set $B \subset \mathbf{C}_2$.

  $\min_B$: let $a \in \mathbf{C}_1$, $b \in \mathbf{C}_2$ and $B \subset \mathbf{C}_2$, then $a \overset{\min_B}{\rightsquigarrow} b$ iff $\forall b' \in B$, $d(a, b) \leq d(a, b')$.

  $\max_B$: let $a \in \mathbf{C}_1$, $b \in \mathbf{C}_2$ and $B \subset \mathbf{C}_2$, then $a \overset{\max_B}{\rightsquigarrow} b$ iff $\forall b' \in B$, $d(a, b) \geq d(a, b')$.

  $\min_{B,k}$: let $a \in \mathbf{C}_1$, $b \in \mathbf{C}_2$ and $B \subset \mathbf{C}_2$, then $a \overset{\min_{B,k}}{\rightsquigarrow} b$ iff $\#\{b' \in B \mid d(a, b) > d(a, b')\} < k$ (i.e., there are less than k entities of B that are closer to a than b is).

- *Range relations* are unordered relations defined between a pair of entities $a \in \mathbf{C}_1$ and $b \in \mathbf{C}_2$, and with respect to a reference value r.

  $<r$: let $a \in \mathbf{C}_1$, $b \in \mathbf{C}_2$ and $r \in \mathbb{R}$, then $a \overset{<r}{\sim} b$ iff $d(a,b) < r$.

  $>r$: let $a \in \mathbf{C}_1$, $b \in \mathbf{C}_2$ and $r \in \mathbb{R}$, then $a \overset{>r}{\sim} b$ iff $d(a,b) > r$.

  Relations $\leq r$ and $\geq r$ can be defined analogously. Notice, anyway, that any range relation can be transformed into an equivalent containment (or element-of) relation as follows: let a, b and r be as before, and let us define

  $$G_{a,r} = \{p \in \mathbb{E}^2 | \mathrm{d(p,a) < r}\}$$

  called the *enlarged* a. Then, we have:

  $a \overset{<r}{\sim} b$ iff $b \overset{\subseteq}{\leadsto} G_{a,r}$;

  $a \overset{>r}{\sim} b$ iff $\neg(b \overset{\subseteq}{\leadsto} G_{a,r})$.

  Relations $< r$ and $\leq r$ can be similarly transformed. Enlarged sets can be difficult to compute for regions. Nevertheless, the above transformation is straightforward if a is a segment or a point: in the latter case, for instance, $G_{a,r}$ is a circle of radius r centered at a.

3 Spatial queries

After defining the relations on spatial entities, we define spatial queries. In our approach, a query is based on a *query object* q, that is an entity from a given class $\mathbf{C}_q$, and a *query space* S, that is a subset of another class $\mathbf{C}_S$. Usually, the query space will be one of the three sets P, L and R defining a *query map* M.

A generic query asks for all entities in the query space that are in a given relation with the query object. More formally, given q and S as above, and a relation $\Re$ defined on $\mathbf{C}_q \times \mathbf{C}_S$, a generic query is denoted by a triple $\prec q, S, \Re \succ$, and it is defined as follows[2]:

$$\textit{return all } s \in S \textit{ that satisfy } q \overset{\Re}{\leadsto} s$$

We will also say that $A = \{s \in S \mid q \overset{\Re}{\leadsto} s\}$ is an *answer* to query $\prec q, S, \Re \succ$, and we will denote $\prec q, S, \Re \succ \mapsto A$.

All spatial queries we consider are defined by proper instantiations of classes $\mathbf{C}_q$ and $\mathbf{C}_S$, and of relation $\Re$ in the generic definition given above. From the classification of relations we have given in the previous Section, we obtain three

[2]This definition is for a generic ordered query; the definition for an unordered query is analogous by substituting $\leadsto$ with $\sim$.

groups of spatial queries; each query directly corresponds to a spatial relation. In the following, we do not give the tedious list of all queries, but we rather discuss the three classes of queries, by focusing on the problems related to their answering, and by giving some relevant examples.

We will denote as before with P, L, and R sets of points, lines, and regions, respectively, and we will denote with p, l, and r a generic point, line, and region, respectively.

3.1 Topological queries

As pointed out in Section 2.1, topological relations are defined on pairs of sets that are topologically consistent; one way to guarantee such consistency is to consider sets from a single map $M = (P, L, R)$. Thus, for a given query, the query space S will be either P, or L, or R, or a suitable subset of one of them, while the query object q will be an entity of M (i.e., $q \in P \cup L \cup R$). We list as examples two region-based and one point-based topological queries, that are often used in geographic applications, together with their informal expression:

$\prec r, R, \overset{RR}{\sim} \succ$: "return all regions of M that are neighbors of r";

$\prec r, L, \overset{RL}{\rightsquigarrow} \succ$: "return the border of region r";

$\prec p, P, \overset{PP}{\sim} \succ$: "return all points that are immediately adjacent to point p".

Notice that the point-based query given above is especially useful for traversing networks, like road maps.

Topological queries can be answered efficiently if map M is represented through a model that encodes topological relations between spatial entities, and if such model is implemented through a data structure that allows efficient retrieval of such relations. In Section 4, we will present and discuss one such model and data structure. We will see that topological queries are in general "easier" than other queries, i.e., the algorithms for retrieving them have a low time complexity.

3.2 Set-theoretic queries

Queries based on set-theoretic relations can have different meanings, and different solutions, when either q belongs to the same map of space S or not.

A set-theoretic query about an entity of the map is generically related with the localization of point or lineal features, and will be solved by using relational operators of the model that we will introduce in Section 4. In other words, if the query object and the query space come from the same map, there exist a structure that "contextualizes" the query object in the space where the answer must be retrieved, and the relational operators incorporated in the model allow an immediate answer (see Section 4.1).

In this Section we will focus on queries induced by set-theoretic relations, when the query object does not belong to the query map. In this case, the contextualization of q must be performed through suitable algorithms that are able to retrieve the entities in S that spatially interfere with q, once q is "plunged" into the query map. For this reason we will also refer to this latter group of queries as *interference queries.*

Coincidence queries are similar to queries in standard databases, as the coordinates of points provide search keys. A coincidence query for points can be easily and efficiently answered if the points in the query space are maintained sorted (e.g., in lexicographic order) in a balanced tree. A coincidence query for lines can be answered analogously, by considering first coincidences of endpoints, and then coincidence of geometries. A coincidence query for regions is slightly more complicated, but it can still be answered in a similar way by scanning the region boundary.

The other queries need suitable algorithms for geometric search, that either work on unstructured data, or are able to exploit the structure of the search space in order to retrieve an answer efficiently. Here, we only list two important queries, corresponding to two basic interference problems: *point location* and *segment intersection.*

$\prec p, R, \overset{\in\mathbf{R}}{\rightsquigarrow} \succ$: "return the region of M containing point p";

$\prec l, L, \overset{\cap\mathbf{LL}}{\sim} \succ$: "return all lines of M intersecting line l".

In Section 5, we will consider the basic techniques offered by computational geometry to search efficiently planar subdivisions (like maps are), and to intersect sets of segments. We will show that all interference queries can be answered through such techniques, combined with the exploration of the topological structure encoding the query map.

3.3 Metric queries

Metric queries are based on metric relations. Thus, they are concerned with the distance between the query object and the entities in the query space. The topological structure of the query map is not much helpful in finding answers to such queries, since the spatial proximity of entities is not necessarily related to their topological adjacency/incidence relations.

In some cases, we can transform metric queries into other non-metric queries: as we anticipated in Section 2.3, range queries can be transformed into interference queries by using enlarged sets. A typical range query is the following:

> $\prec p, P, \stackrel{<r}{\sim} \succ$: "return all points of P that lie closer than a distance r from point p".

Such a query can be substituted with the equivalent query:

> $\prec G_{p,r}, P, \stackrel{\ni}{\rightsquigarrow} \succ$: "return all points of P inside region $G_{p,r}$",

where region $G_{p,r}$ is the enlarged set of p, i.e., a circle of radius r centered at p.

Unfortunately, there is no way to cheat metric problems that arise with min/max queries, which are, however, very important in the context of spatial databases. A typical example of min query is the following:

> $\prec p, P, \stackrel{\min_P}{\rightsquigarrow} \succ$: "return the point in P that lies closest to point p".

Auxiliary topological structures superimposed on the query map, like Voronoi diagrams, can help answering the above query; on the other hand, such structures are useful only in some cases, while they require a considerable amount of storage [12].

In order to answer min/max queries efficiently, we would like to have spatial information encoded into models that are more space-based than object-based. On the other hand, space-based structures do not allow an explicit encoding of spatial entities, and, especially, they do not offer any support for encoding topological relations, that are the basis of a large part of spatial queries. Thus, if we want to maintain a topological object-centered model for spatial entities, while guaranteeing efficient spatial search through space-based structures, we must integrate the two approaches.

A solution consists of using *spatial indexes*, like *grids* or *quadtree-based structures*, as superstructures on the topological model encoding the maps. As spatial indexes allow efficient spatial search, they are not only essential for answering metric queries, but they can also help speeding-up the treatment of other spatial queries, and of operations like map overlay, that we will consider later. Finally, spatial indexes help to organize information for efficient disk storage. In Section 6, we will discuss the use of spatial indexes, and their integration with the topological model presented in Section 4.

3.4 Multiple queries

It often happens that a query induced by the same relation $\Re$ is made on the same query space S for many query objects from a set Q, that we call the *query set*: a typical example is when we want to classify a whole set of entities from a given map with respect to another map. A *multiple query* is defined by considering all queries about entities of Q as a whole, as follows:

$$\forall q \in Q,\ \textit{return all } s \in S \textit{ that satisfy } q \stackrel{\Re}{\leadsto} s.$$

We will denote the multiple query defined above by $\prec Q, S, \Re \succ$.

A multiple query can be treated either as a "set of queries", or as a "query about a set". The former approach is trivial, and the corresponding solution is to answer independently to each query about an object $q \in Q$. The latter approach considers using some global technique that, given Q, S, and $\Re$, is able to answer to $\prec Q, S, \Re \succ$ at a reduced computational cost, with respecto to the time needed when considering each entity independently.

The trivial approach is convenient whenever the cost of answering a single query is of the same order of the size of its output: the cost of answering a multiple query cannot be smaller than the size of the global output, that is equal to the sum of all sizes of outputs of single queries, i.e., to the cost of answering all single queries. Thus, we cannot hope to achieve better results with the global approach. For instance, all topological queries can be answered efficiently in this way.

In other cases, especially for interference queries, a single query has a cost whose order is higher than the output size, being a function $f(n)$ (either logarithmic or linear) of the size n of S. Solving single queries independently gives a total cost of $m\,f(n)$, where m is the cardinality of Q. We will see in Section 5 how, depending on the relation $\Re$ inducing the query, we can use suitable algorithms to achieve a better performance under the global approach. A typical example of such queries is multiple line intersection, e.g., intersection between two networks.

A global approach based on sorting can be used to answer coincidence queries: if objects in Q and S are maintained sorted according to some key (e.g., lexicographically, for points), a multiple coincidence query can be answered in time $O(\min(m,n))$ by scanning the sorted lists in parallel, while the trivial approach would require at least $O(m\log(n))$ time.

Also multiple metric queries, that could be answered by using spatial indexes, can be faced efficiently with a pseudo-global approach: spatial buckets often correspond to disk pages; if entities in the query set are sorted according to the spatial bucket containing each entity, single queries involving search in the same spatial region can be grouped, and disk access is thus minimized.

3.5 Overlays

Most geographic databases organize data in several *thematic maps*: information on entities contained in a given portion of space are spread over different maps, depending on their semantics. An *overlay* of maps consists in taking two or more such maps, and producing a new map, that is a fusion of a part or all the information contained in the input maps.

More formally, given two maps $M' = (P', L', R')$ and $M'' = (P'', L'', R'')$, their overlay $M' \oplus M''$ will be a map consisting of:

- all points of $P' \cup P''$ plus all intersections between lines of L' and L'';
- all lines obtained by breaking the lines of $L' \cup L''$ at intersection points;
- all regions obtained by fragmenting the regions of $R' \cup R''$ with the network of lines above, i.e., all maximal portions of space that have the property of being completely contained into one region of R' and into one region of R''.

Such new entities will be topologically related among them, and the new map could be included in the database if a suitable model is built that represents it.

Windowing, i.e., the extraction of a portion of map contained inside a given (usually rectangular) window is, at least geometrically, a special case of an overlay, where one of the two maps is trivially the window itself.

While the aim of the spatial queries is to report entities (contained in the database), the aim of overlays is to build new entities. It is not our purpose here to make a thorough discussion of all aspects and problems related with overlay operations. Nevertheless, we have introduced overlays in this Section, because the geometric problems involved in their computation (essentially, region and segment intersection, and point classification), are the same met when facing multiple interference queries. Thus, algorithms for answering multiple interference queries are also useful to solve overlays (see Section 5).

4 A topological model for maps

As already mentioned in the introduction, the problem of representing spatial entities, both in 2D and in 3D, together with their relations in a comprehensive and global model has been faced by several authors, in the fields of geometric modeling, computational geometry, and GIS.

In the 2D case, most literature in geometric modeling and computational geometry is concerned with the representation of *planar subdivisions*, i.e., partitions of a planar domain into simply connected regions, induced by a *Plane Straight Line Graph (PSLG)* [24, 13, 31]. A PSLG is a connected Euclidean graph with non-crossing straight-line edges and no dangling chains, i.e., such that every vertex has at least degree two. A PSLG Σ partitions a portion of the Euclidean plane into a set of simply connected regions, bounded by edges of Σ and containing no edge or vertex of Σ in their interior. Thus, points and lines in planar subdivisions are introduced only as parts of the boundaries of regions, while no isolated points or open chains are allowed.

In geographical information theory, some authors have proposed models for 2D maps based on *simplicial complexes*, that are special cases of PSLGs with triangular facets [10, 6, 33]. Some other authors have pointed out the need for models that overcome the limits of PSLGs by encoding also point and lineal features without introducing dummy entities [22, 23]; this trend is present on a more general basis in recent research in geometric modeling [26, 27].

The model we propose here extends a planar subdivision by allowing the graph inducing the subdivision to be any planar graph with piecewise linear edges, and possibly containing isolated points and/or dangling chains. The result is similar in principle to the *single valued vector maps* proposed in [22], but here we focus on the explicit encoding of all spatial entities, and mainly on the embedding of relational operators in the model. These requirements lead to the development of an efficient data structure for representing the model, and of efficient accessing algorithms that encode relational operators.

4.1 The Plane Euclidean Graph

The *Plane Euclidean Graph (PEG)* is defined by an Euclidean graph $G = (V, E, F)$, where:

- V, called the set of *vertices*, is a set of points in the plane;
- E, called the set of *edges*, is a set of polygonal chains having their endpoints in V, and such that any two edges of E never cross (i.e., they never intersect, except at their endpoints). A *polygonal chain* of E is defined as a sequence of points $e = (v, p_1, \ldots, p_k, w)$, where $v, w \in V$ and $p_1, \ldots, p_k \notin V$; points $p_1, \ldots, p_k$ are called *joints* of e.
- F, called the set of *faces*, is a set of maximal regions f bounded by chains of E, such that for every two points P and Q inside f, there exists a curve on the plane that joins P to Q without intersecting any edge of E.

In practice, a PEG is induced by any Euclidean graph with piecewise linear edges, with the only restriction of being a plane graph. The planarity is necessary to guarantee that topological relations between entities in the PEG are well defined.

Given a map $M = (P, L, R)$, defined as in Section 2, we can represent M through a PEG $G_M = (V, E, F)$ where $V \equiv P$, $E \equiv L$, and $F \equiv R$; this notation is intended to make a distinction between spatial entities and their representations embedded in the topological model. In a PEG:

- A vertex of degree zero represents a point feature.
- A vertex of degree one represents an endpoint of a lineal feature.
- A vertex of degree two represents a junction point, either on the border between two regions or on a lineal feature: as edges are polygonal chains, junction points should not be introduced (unless they represent point features lying on edges) in order to make the topology of the model as simpler as possible. Notice that a closed chain that does not contain relevant points can be represented, without introducing vertices, as a sequence of joints, where the first and the last joints are coincident.
- A vertex of degree more than two represents either a branch point or a cross point. Besides being necessary in order to ensure the planarity of the PEG, branch and cross vertices are relevant to the semantics of the map.
- An edge completely contained inside a face represents a lineal feature.
- An edge that separate two regions represents a border edge.

Entities in a PEG G are topologically related, like the spatial entities they represent, according to the definitions given in Section 2.1. Moreover, entities representing point and lineal features of the map are related to edges and faces of the model through element-of and containment relations given in Section 2.2.

We have defined *relational operators* on the PEG, capable of returning objects topologically related with a given entity of the model. Relational operators directly reflect the three classes of adjacency, boundary, and co-boundary relations we introduced in Section 2.1. Let $G = (V, E, F)$ be a PEG, and let $v \in V$, $e \in E$, and $f \in F$ generic elements of G. We define on G nine topological operators as follows:

- $VV(v)$: returns all vertices of V that are adjacent to v in G;
- $VE(v)$: returns all edges of E that are part of the co-boundary of v in G;
- $VF(v)$: returns all faces of F that are part of the co-boundary of v in G;
- $EV(e)$: returns the two vertices of V that are endpoints of e in G;
- $EE(e)$: returns all edges of E that are adjacent to e in G;

- $EF(e)$: returns the two faces of F that form the co-boundary of e in G;
- $FV(f)$: returns all vertices of V that are part of the boundary of f in G;
- $FE(f)$: returns all edges of E that are part of the boundary of f in G;
- $FF(f)$: returns all faces of F that are adjacent to f in G.

Notice that the above operators directly answer topological queries we defined in Section 3.1. Moreover, we define the following set-theoretic operators, that encode relations involving point and lineal features af the map. Such operators directly answer to set-theoretic queries when the query object is part of the query map (the intersection operator is not given because intersections are not allowed within a PEG):

- $\in(v)$: returns the face of F containing point feature v;
- $\subset(e)$: returns the face of F containing lineal feature e;
- $\ni(f)$: returns all point features of V belonging to face f in G;
- $\supset(f)$: returns all lineal features of E contained into face f in G.

In Section 5, we will consider geometric algorithms for solving interference queries that manipulate PEGs. As such algorithms always deal with straight-line segments, and regions bounded by such kind of lines, we introduce here the *expanded graph* of a PEG. The idea is simply to divide each edge (polygonal chain) of the PEG into the straight-line segments forming it, and to add these segments to the set of edges, and their endpoints (corresponding to joints) to the set of vertices. Formally, given a PEG G, we define its *expanded graph* $G^* = (V^*, E^*, F)$ as follows:

- $V^* = V \cup \{p \mid p \text{ is a joint of } e \text{ for some } e \in E\}$;
- $E^* = \{(p, q) \mid \exists e \in E$ such that p and q are either vertices or joints with consecutive positions on $e\}$.

Notice that the faces of a PEG and of its expanded graph are coincident.

4.2 A data structure for the PEG

In order to encode a PEG efficiently, we need to define a data structure that fulfills the following requirements:

1. all entities of a graph $G = (V, E, F)$ are explicitly encoded;

2. a part of the relations used by the relational operators are explicitly encoded;

3. all other relations can be retrieved efficiently, i.e., all relational operators can be computed in a time that is linear in their output size.

4. the storage space is as small as possible.

The data structure we propose for representing a PEG is a modification of the well known DCEL data structure [24], extensively used for encoding planar subdivisions. The data structure is edge-oriented, as it encodes explicitly relations between each edge and its either adjacent, or boundary, or co-boundary entities. A specification of the data structure follows:

- For each vertex:
  - its two coordinates;
  - if the vertex is isolated, a pointer to the face containing it, otherwise, a pointer to one of the edges incident into it;
- For each edge:
  - a pointer to its geometry (a chain of joints);
  - two pointers to its endpoints (vertices);
  - two pointers to its incident faces (in case the edge is a lineal feature, the two faces will be coincident);
  - four pointers to the first adjacent edges met by rotating counterclockwise and clockwise about its endpoints, respectively.
- For each face:
  - a pointer to a list of edges, containing one edge for each connected component of its boundary;
  - a pointer to a list of edges, containing one edge for each connected component of its contained edges (lineal features);
  - a pointer to a list of isolated points contained in the face.

The list of vertices can be maintained sorted (e.g., lexicographically), and stored into a balanced tree; also, similar lists of edges and vertices could be maintained.

It is easy to show that such data structure fulfills the requirements stated above. A complete analysis of its space complexity and of the time complexity of relational operators implemented on it is omitted here, for brevity.

This data structure is intended for main memory; disk storage, instead, requires partitioning information among different disk pages. It is not immediately clear, and it is not our subject here, how to distribute entities such that links between them can be retrieved efficiently. Spatial indexes we discuss in Section 6 can be used to group together into buckets entities that are spatially close one another.

5 Algorithms for interference queries

In this Section, we briefly outline how geometric problems that arise in answering interference queries can be solved by using efficient algorithmic techniques developed in computational geometry.

Interference queries induced by element-of, containment and intersection relations can be answered efficiently by solving the point location problem, mentioned in Section 3.2. Let G be a PEG representing a map, let G^* be its expanded graph, and let n be the number of non-isolated vertices of G^*. Then, the location of a query point p in G would require $O(n)$ time by using a "brute force" approach (exhaustive search). A more interesting approach based on a preprocessing of the query map is the so-called *slab method* [24]. The expanded PEG G^* is intersected with $n+1$ horizontal strips obtained by drawing a horizontal straight-line through each of its n vertices. The intersection of G^* with each slab defines a set of non-intersecting segments, thus partitioning G^* into trapezoids. Slabs are sorted by y-coordinate (with $O(n \log n)$ preprocessing time), and the slab containing the query point p can be retrieved in $O(\log n)$ time by applying a binary search. Similarly, trapezoids within a slab are sorted along the x-axis, and the trapezoid (or the segment) containing p can be retrieved in $O(\log n)$ time by applying another binary search within the slab. The trapezoidal diagram can be computed in $O(n^2)$ preprocessing time, but its main drawback is an $O(n^2)$ storage cost.

Other methods for point location have been proposed in the literature, that require linear storage cost. Among them, we can mention the *chain method* [20], that is based on the decomposition of a G^* into a set of monotone chains; point location can be performed in $O(\log^2 n)$ time, using $O(n)$ storage and with $O(n \log n)$ preprocessing time. Another method, which also achieves $O(\log n)$ query time, is the *triangulation refinement method* [19], that requires $O(n)$ storage and $O(n \log n)$ preprocessing time.

The segment intersection problem defined in Section 3.2 would require again $O(n)$ processing time, if solved through exhaustive search. The performance can be improved by first applying a point location algorithm to the endpoints of the query segment l ($O(\log n)$ time), and then traversing G^* while testing the intersections of l only with the edges bounding (or contained into) the faces crossed by l.

A different approach can be used for solving multiple interference queries, involving multiple line intersection. This approach is based on a sweep-line technique, and it can also be used successfully for computing overlays [1]. Let $G_1^* = (V_1^*, E_1^*, F_1)$ and $G_2^* = (V_2^*, E_2^*, F_2)$ be two expanded PEGs with n and m vertices, respectively. The endpoints of the edges of E_1^* and E_2^* are first sorted by x-coordinate (with $O((n+m)\log(n+m))$ preprocessing time).

Then, a vertical sweep-line is moved over the $n+m$ segments from left to right, stopping at each endpoint and at each intersection point between two lines. For each such event, an ordered collection of the line segments intersected by the sweep-line is maintained, called the sweep-line status. At each event, the algorithm performs one of the following operations, depending of the kind of event:

- the left endpoint of a new segment s is met: the new segment s enters the sweep-line status, that is updated accordingly;
- the right endpoint of a segment s is met: the segments leaves the sweep-line status, that is updated accordingly;
- an intersection point between two segments $s_1 \in E_1$ and $s_2 \in E_2$ is met: s_1 and s_2 are exchanged in the sweep-line status.

At each event, all the pairs of segments that become adjacent in the status are checked for possible intersections. If an intersection is found, it is added in the sorted list of events. The event list and the sweep-line status are maintained into two dynamic structures, a priority queue and a balanced tree, respectively. This leads to a global time complexity of $O((k+n+m)\log(n+m))$, where k is the number of intersections.

More recently, an optimal algorithm has been proposed [4], which achieves an $O((n+m)\log(n+m)+k)$ time complexity. Although the crucial tool is still the sweep-line, not only the status but the entire scene to the right of the sweep-line is maintained. In order to reach the optimal computational cost, refined tools (such as segment tree, topological sweep and cost amortization) are used. Unlike the suboptimal algorithm presented above, which uses $O(n+m)$ storage, this technique uses $O(n+k)$ space. Therefore, it remains an open question whether optimal time and space performances are simultaneously attainable.

6 Spatial indexes

The topological model we have defined in Section 4 gives an object-based representation of maps, that offers useful support to the solution of several problems related to spatial queries and overlays. On the other hand, as already pointed out, for some purposes, it would be convenient to organize spatial data according to a space-based scheme, where objects having close locations in space are grouped together. Major reasons for adopting a space-based scheme are the following:

1. answering metric queries efficiently;
2. speeding-up interference queries and overlays by localization of interference computation;
3. achieving efficient disk storage and retrieval.

It is not our purpose to discuss here the third point, that would deserve a separate and thorough study. Our aim is to obtain a *hybrid model*, that is able to support both an object-based and a space-based approach to the representation of spatial entities, by superimposing a spatial index to the topological model. Approaches based on hybrid structures have been proposed also in three dimensions [2, 28, 3].

A *spatial index* is essentially a space partitioning technique that allows locating efficiently any spatial entity in the portion of space it occupies. The simplest and most popular example of a spatial index is the *regular rectangular grid*, that partitions a rectangular domain using equally-spaced rectangular cells. A point can be located in constant time, while a line or a region can be located in a time that is linear in the number of cells it crosses. A hierarchical approach to spatial indexing is the *quadtree*, where space is recursively subdivided into quadrants. In this case, the quadrant containing a point can be found in a time that is linear in the height of the quadtree, i.e., logarithmic in the maximum number of its quadrants. In the following we will refer to both a cell of a grid and a quadrant of a quadtree with the generic term of *bucket*.

The main difference between using buckets in the context of a hybrid model and following a raster approach in encoding spatial data, is that here buckets are not atomic entities: a bucket is rather a container of (parts of) spatial entities, that are fully described in the topological model. Suitable links must be set between the spatial index and the topological model, in order to support efficient retrieval of entities contained or crossing a given bucket.

If a regular grid is used, there is a priori no bound on the maximum number of entities per bucket: with a coarse grid, a large number of entities per bucket is expected; conversely, with a fine grid, a large number of almost empty buckets is expected. Quadtree based structures, like PM-quadtrees, PMR-quadtrees, R-trees, and R_+-trees [28], allow a control over the maximum number of spatial entities per bucket by refining the spatial index only where the density of entities is higher. PM-quadtrees are well-suited as spatial indices for maps, since they allow exact representation of point, line, and region data; PMR-quadtrees are especially well-suited for networks, since they are designed to represent line data.

R-trees and R_+-trees are based on a binary partitioning technique, they allow the representation of generic geometric objects, and they are especially suited for mapping the spatial index on disk, because their structure reflects the B-tree structure of disk pages.

Independently of the spatial index used, queries and overlays are solved by localizing computations and searches into buckets, provided that efficient retrieval of spatial entities in a bucket is supported. In the following, we discuss the major techniques adopted. For brevity, we consider the main classes of problems, and we give, for each class, a relevant example.

Metric queries are solved by searching first the bucket(s) containing the query object, and then searching its adjacent buckets, if necessary. Let p be a query point, P be a query space of points, and $\prec p, P, \overset{\min_P}{\leadsto} \succ$ be a min query for p with respect to P.

1. The bucket B containing p is found, and the set P_B of points of P lying in B is retrieved.
2. An exhaustive search is performed on P_B to find the point $p' \in P_B$ that is closest to p.
3. If either P_B is empty or the circle centered at p and through p' intersects adjacent buckets that have not been visited yet, points 2 and 3 are repeated on such adjacent buckets.
4. Otherwise p' is the answer.

Such a technique is explained in details in [17] for metric search based on PMR-quadtrees.

Interference queries are improved simply by localizing computation into buckets crossing or containing the query object. Let l be a query line, L be a query space of lines, and $\prec l, L, \overset{\cap \mathbf{LL}}{\sim} \succ$ be an intersection query for l on L.

1. All buckets $B_1, \ldots, B_k$ intersected by l are found, and all corresponding subsets $L_{B_1}, \ldots, L_{B_k}$ of L are retrieved.
2. An intersection algorithm is applied for each subset independently.

Multiple interference queries can be solved similarly by first selecting all buckets crossing the query objects, and then applying the suitable geometric algorithm inside each bucket. If the number of query objects is large enough, i.e., if most bucket are expected to be involved, it is better to apply the following technique described for overlays.

Overlays are improved by scanning all buckets and solving the problem locally inside each bucket, as above. Let M' and M'' be two maps; we want to compute $M' \oplus M''$. For each bucket B in the space covered by the two maps do:

1. Retrieve the entities M'_B and M''_B of the two maps crossing bucket B.
2. Solve $M'_B \oplus M''_B$.

Experiences of efficient overlays on large maps using regular grids are discussed in [11].

7 Concluding remarks

We have introduced a formal framework for two-dimensional spatial entities and we have classified three major groups of spatial relations on such entities: topological, set-theoretic, and metric relations. On the basis of spatial relations, we have formally defined spatial queries, and we have discussed geometric problems related to their solution.

We have presented a topological model for representing spatial entities forming maps and their topological relations, and we have discussed how such model supports topological queries. We have reviewed major geometric algorithms for interference queries and overlays. Finally, we have discussed how metric queries can be supported, and interference queries and overlays can be improved by using a spatial index. The integration of a topological model with a spatial index provides a hybrid model that supports both an object-based and a space-based description of two-dimensional geometric entities.

Spatial databases should be built on models that support effective and efficient representations of spatial data and their relations. Moreover, they should incorporate suitable algorithms that allow efficient information retrieval for spatial queries. Our work is a step towards the design of one such database for geographic maps.

Our framework for the formal definition of spatial entities, relations, and queries can be extended to the three-dimensional case for applications like solid object representation in a CAD system [3]. Also, suitable topological models, spatial indexing structures, and geometric algorithms are known in the literature, that can help solving representation and retrieval problems in the three-dimensional case by following our approach.

The efficient encoding of spatial entities on disk is a very important subject in spatial database design, that we plan to tackle in the future in order to implement our approach. Also, the integration of spatial and tabular data in an environment that supports both the structure and the attributes of spatial entities, together with standard alphanumeric information, should be faced.

References

[1] Bentley, J.L, Ottmann, T.A., "Algorithms for reporting and counting geometric intersections", *IEEE Transactions on Computers*, 28, pp. 643-647, 1979.

[2] Brunet, P., Navazo, I., "Solid representation and operation using extended octrees", *ACM Transaction on Graphics*, 8, 1989.

[3] Bruzzone, E., De Floriani, L., Pellegrinelli, M., "A hierarchical spatial index for cell complexes", in *Proceedings 3rd International Symposium on Large Spatial Data Bases*, Singapore, June 1993, (in print).

[4] Chazelle, B., Edelsbrunner, H., "Optimal solution for intersecting line segments", in *Proceedings 29th IEEE Symp. on Foundation of Computer Science*, October 1988.

[5] Dobkin, D.P., Laszlo, M.J., "Primitives for the manipulation of three-dimensional subdivisions", *Algorithmica*, 5(4), pp.3-32, 1989.

[6] Egenhofer, M., Frank, A.U., Jackson, J.P., "A topological model for spatial databases", *Lecture Notes in Computer Science*, N.409, pp.271-286, 1989.

[7] Egenhofer, M., "A formal definition of binary topological relationships", *Lecture Notes in Computer Science*, N.367, pp.457-473, 1989.

[8] Egenhofer, M., Herring, J., "A mathematical framework for the definition of topological relationships", in *Proceedings 4th International Symposium on Spatial Data Handling*, pp.803-813, Zurich, Switzerland, July 1990.

[9] Egenhofer, M., Franzosa, R., "Point-set topological spatial relations", *International Journal of Geographical Information Systems*, 5(2), pp.161-174, 1991.

[10] Frank, A., Kuhn, W., "Cell graph: a provable correct method for the storage of geometry", *Proceedings 2nd International Symposium on Spatial data Handling*, seattle, WA, 1986.

[11] Franklin, W.R., et al., "Uniform grids: a technique for intersection detection on serial and parallel machines", in *Proceedings Auto Carto 9*, Baltimore, MD, April 2-7, 1989, pp. 100-109.

[12] Gold, C.M., "The meaning of "neighbour" ", *Lecture Notes in Computer Science*, N.639, Springer-Verlag, 1992, pp. 220-235.

[13] Guibas, L., Stolfi, J., "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams", *ACM Transactions on Graphics*, 4(2), pp.75-123, 1985.

[14] Günther, O., *Efficient Structures for Geometric Data Management* (LNCS 337), Spriger-Verlag, 1987.

[15] Herring, J., "TIGRIS: topologically integrated GIS", *Proceedings Autocarto 8*, ASPRS/ACSM, Baltimore, MD, pp.282-291, March 1987.

[16] Herring, J., Egenhofer, M.J., Frank, A.U., "Using category theory to model GIS applications", in *Proceedings 4th International Symposium on Spatial Data Handling*, pp.820-829, Zurich, Switzerland, July 1990.

[17] Hoel, E.G., Samet, H., "Efficient processing of spatial queries in line segment database", *Lecture Notes in Computer Science*, N.525, Springer-Verlag, 1991.

[18] Kainz, W., "Spatial relationships - Topology versus order", in *Proceedings 4th International Symposium on Spatial Data Handling*, pp.814-819, Zurich, Switzerland, July 1990.

[19] Kirkpatrick, D.G., "Optimal search in planar subdivision", *SIAM Journal of Computing*, 12(1), pp. 28-33, 1983.

[20] Lee, D.T., Preparata, F.P., "Location of a point in a planar subdivision and its applications", *SIAM Journal on Computing*, 6(3), pp. 594-606, 1977.

[21] M. Mäntylä, *An Introduction to Solid Modeling*, Computer Science Press, Rockville, MD, 1987.

[22] Molenaar, M., "Single valued vector maps - a concept in GIS", *Geo-Informationssysteme*, Vol.2, No.1, 1989.

[23] Pigot, S., "A topological model for a 3D spatial information system", *Proceedings 5th International Symposium on Spatial Data Handling*, Charleston, SC, August 3-7, 1992.

[24] Preparata, F.P., Shamos, M.I., *Computational Geometry: an Introduction*, Springer-Verlag, 1985.

[25] Requicha, A.A.G., Voelcker, H.B., "Solid modeling: a historical summary and contemporary assessment", *IEEE Computer Graphics and Applications*, 2, 2, pp.9-24, 1982.

[26] Rossignac, J.R., O'Connor, M.A., "SGC: a dimensional-independent model for pointsets with internal structures and incomplete boundaries", *Geometric Modeling for Product Engineering*, Wosny, M.J., Turner, J.U., and Preiss, K., Eds., Elsevier Science Publishers B.V. (North Holland), pp. 145-180, 1990.

[27] Rossignac, J.R., "Through the cracks of the solid modeling milestone", *Eurographics 91 State of the Art Report on Solid Modeling*, pp. 23-109, 1991.

[28] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[29] Smith, T.R., Park, K.K., "Algebraic approach to spatial reasoning", *International Journal of Geographical Information Systems*, 6(3), pp.177-192, 1992.

[30] Whitney, H., *Geometric Integration Theory*, Princeton University Press, 1957.

[31] Woo, T.C., "A combinatorial analysis of boundary data structure schemata", IEEE Computer Graphics and Applications, 5, 3, pp.19-27, 1985.

[32] Worboys, M.F., Hearnshaw, H.M., Maguire, D.J., "Object-oriented data modelling for spatial databases", *International Journal of Geographical Information Systems*, 4(4), pp.369-383, 1990.

[33] Worboys, M.F., "A generic model for planar geographic objects", *International Journal of Geographical Information Systems*, 6(5), pp.353-372, 1992.