

Modeling Topological Spatial Relations: Strategies for Query Processing

Eliseo Clementini*, Jayant Sharma†, and Max J. Egenhofer§

National Center for Geographic Information and Analysis,
University of Maine, Orono, ME 04469-5711, U.S.A.

Abstract

This paper investigates the processing of spatial queries with topological constraints, for which current database solutions are inappropriate. Topological relations, such as *disjoint*, *meet*, *overlap*, *inside*, and *contains*, have been well defined by the 9-intersection, a comprehensive model for binary topological relations. We focus on two types of queries: (1) “Which objects have a stated topological relation with a given spatial object?” and (2) “What is the topological relation between two given spatial objects?” Such queries are processed at two levels of detail. First, Minimum Bounding Rectangles are used as an approximation of the objects’ geometry and as a means of identifying candidates that might satisfy the query. Next, the nine intersections that determine the topological relations between candidate pairs are calculated. We present algorithms for minimizing these computations. Considerable performance can be gained by exploiting the semantics of spatial relations. We also compare the approach for a *naive cost model*, which assumes that all relations have the same frequency of occurrence, with a *refined cost model*, which considers the probability of occurrence of the topological relations. The strategies presented here have three key benefits: (1) they are based on a well-defined formalism; (2) they are customizable; and (3) they can take into account important statistical information about the data.

1. Introduction

Geographic Information Systems (GISs) contain high-level spatial operators that are uncommon in conventional database management systems (DBMSs) [1]. Spatial operators appear, for instance, as constraints in spatial queries to select spatial objects. They may include such simple selections as, “Retrieve all lakes *in* the state of Maine,” or more complex ones like, “Find the *shortest path* from Boston to Bangor based on travel time.” Traditionally, concerns about processing spatial queries have been addressed primarily at the level of *spatial access methods* in order to minimize the number of disk accesses by clustering spatial objects according to their spatial neighborhoods. This method supports requests like, “Display a map of Maine showing highways and cities with a population of more than 25,000” where the constraint is that objects intersect with a search window. For other types of queries in which the semantics of the constraints are more complex, the mere support of spatial access methods is insufficient to guarantee efficient processing of queries.

Spatial access has to be complemented by methods that consider (1) the semantics of the spatial relations, i.e., how the relations are defined, (2) heuristics for evaluating spatial constraints, and (3) estimates of the distribution, i.e., the probability of occurrence of the relations. The *semantics of the spatial relations* allow for the inference of a relation from a set of given relations. For example, given that region *A* is *disjoint* from region *B* and that *B contains* region *C*, it can be inferred that *A is disjoint* from *C* [2]. *Heuristics* for evaluating spatial constraints are dependent on

* This work was performed while on a leave of absence from Università di L’Aquila, Dipartimento di Ingegneria Elettrica, 67040 Poggio di Roio, L’Aquila, Italy. Eliseo Clementini is partially supported by the Italian National Council of Research (CNR) under grant No. 92.01574.PF69.

† Jayant Sharma receives partial support from a University of Maine Graduate Research Assistantship and the NCGIA.

§ Max Egenhofer’s work is partially supported by NSF grant No. IRI-9309230, a grant from Intergraph Corporation, a University of Maine Summer Faculty Research Grant, and the NCGIA through NSF grant no. SBR-8810917.

the spatial data model and data structures used. An example of a heuristic would be the use of Minimum Bounding Rectangles (MBRs) as a first approximation of the objects' geometry as a fast filter. *Estimates of the distribution* are important, because the application often determines what relations are feasible. For example, in a cadastral application the only possible topological relations between land parcels are *disjoint* or *meet*.

This paper focuses on the processing and algebraic optimization of spatial queries with *topological constraints*. An example of such a query is, "Find all residential lots for sale *adjacent* to Branch Lake," where *adjacent* is a topological relation. Such relations are usually not explicitly stored among spatial objects, but have to be inferred from the objects' geometry. For example, the fact that two land parcels are *adjacent* would be inferred from the fact that the two regions have a part of their boundaries, but no interior, in common. While existing DBMSs do not support such complex relations, extensible DBMSs [3] have the provisions to incorporate them into query languages. To be successful as geographic databases, extensible DBMSs need models of how to process and optimize queries over spatial relations.

The query processing strategies presented in this paper are based on the 9-intersection, a comprehensive model for binary topological relations among point-, line-, and area-objects [4, 5]. It identifies eight basic relations between two regions in \star' ; 19 topological relations between a region and a simple line in \star' ; and 33 relations between two simple lines in \star' . The strategies for processing queries with topological constraints are based on the observation that only a true subset of the nine intersections need to be determined in order to identify the topological relation between two spatial objects. For example, determining whether two regions are *disjoint* only requires determining that two intersections, the boundary-boundary and the interior-interior, are empty because in none of the other seven possible cases are these two intersections both empty.

The optimization of topological queries is particularly challenging because terminology and semantics of the relations varies across application domains. As long as no formalizations of such spatial predicates exist that would match with humans' interpretations, it is standard practice to define the set of topological relations for an application domain as disjunctions of a set of basic, mutually exclusive relations [6]. The sets of relations defined by the 9-intersection form base sets, from which database users and administrators can construct the non-primitive sets of relations relevant to their specific application domain. For example, an application domain may not need the distinction between *covers* and *contains* as defined in the 9-intersection model for region-region relations; therefore, for this user group, the set of relevant region-region relations would be $\{disjoint, containsOrCovers, insideOrCoveredBy, meet, equal, overlap\}$. It is important that any query optimization strategy selected will work independently of the particular combinations made.

This general applicability is of great importance since initial tests with human subjects clearly demonstrate that humans group conceptually close relations and devise a prototypical representative of this group [7]. Two topological relations are conceptually close if there is a transition from one to another as a result of a gradual deformation applied to one object [8, 9]. Given this evidence it is unlikely that all users would need or grasp the nuances between the 19 distinct region-line or 33 line-line relations in their applications. The particular subset, or subgroupings, of these relations will necessarily vary with the application domain. Hence any query processing strategy based on the base sets of relations must be applicable to such domain specific groupings. In this paper we give examples of how the strategies presented can be directly applied to such cases, and therefore, these strategies provide a degree of flexibility and customizability not typically found in conventional spatial query processors.

We assume an object-centered geographic database. Object-centered geographic databases [1] have a vector or topological data model and deal with spatial objects that have a distinct identity, e.g., in the form of simplicial complexes or cells [10, 11]. This spatial data model represents the geometry of geographic objects in terms of points, lines, and areas, and records explicitly boundary and co-boundary relations among the geometric elements.

There are two types of queries, the processing of which requires the computation of the values of the nine intersections between the interiors, boundaries, and exteriors. They are:

- “Find all objects that have the topological relation R to object A ?” and
- “What is the topological relation between objects A and B ?”

The latter type of query is less frequently asked though such a query is as important as the former in geographic applications. The results of these queries have been called “qualitative answers” [12].

The goal of this paper is to present the most promising strategies for processing topological queries. The novel result is an algorithm to determine the smallest subset of the nine intersections that have to be evaluated. We demonstrate that such spatial queries can be frequently determined with less effort than computing all nine intersections for a topological relation. We start with a “naive” model of query processing, assuming that the computation of all intersections is of equal complexity, that all relations are equally distributed, and that all relations are equally frequently queried. The naive model is then refined by assigning a probability distribution to the set of relations. For example, in a cadastral application the most frequent topological relation is “disjoint” followed by “meet.”

The framework introduced in this paper allows a query processor to find the best strategy to assess a topological relation between two spatial entities. The 9-intersection model gives the primitives for describing a topological relation, and by using a combination of such primitives it is possible to model every set of topological relations. The same elementary tools can be used even when such complex geographic objects as regions with holes or 1-spheres are involved [13].

The remainder of this paper is structured as follows: after a brief summary of the pertinent work in spatial query optimization, we review the concepts of the 9-intersection as the model for which we will investigate query processing strategies. As a first step, we introduce the mappings from topological relations as defined by the 9-intersection onto relations between MBRs and show how such knowledge can be exploited as a fast filter to find candidates that would satisfy a particular topological relation. Subsequently, we design algorithms to select objects from the candidates for two types of spatial queries: (1) finding the set of objects that hold a particular topological relation, or set of topological relations, with respect to a given object; and (2) determining the topological relation between two given objects. For the latter we compare the approach for the *naive cost model* with a *refined cost model*.

2. Previous Work in Spatial Query Optimization

Most approaches to spatial query processing reported in the literature optimize spatial queries by transforming user queries into evaluation plans that take into account the extended physical storage mechanisms and access methods. This section reviews four such approaches and compares them with the goals of this paper.

2.1 Spatial and Non-Spatial Database (SAND)

Aref and Samet [14, 15] describe strategies for constructing query evaluation plans and assessing their costs. SAND consists of separate spatial and non-spatial data stores, which have two-way links between records that describe the spatial and non-spatial attributes of some object. The strategies essentially extend traditional non-spatial approaches with a spatial selection, which retrieves records from the spatial data store that satisfy the given constraints. The evaluation plans involve reordering the selections, i.e., choosing between performing the spatial or non-spatial selection first; using indexes on both spatial and non-spatial data; and performing spatial operations while accessing the data rather than storing links in a temporary store and subsequently traversing these records to perform the desired operations. The major contribution of SAND is its extensibility because the techniques are applicable to various data types, not only to spatial data.

2.2 GEOQL

Another proposal for extending a traditional DBMS and query processor uses an augmented SQL, called GEOQL, which has spatial operators and works with an extended SQL-based DBMS [16]. Queries are optimized in four stages. (1) A logical transformation removes redundant constraints and builds a query tree such that spatial indexes can be effectively utilized; (2) decomposition partitions the tree such that spatial and non-spatial subqueries are formed; (3) from this set of subqueries various execution plans are formulated and evaluated, one of which is executed; and (4) the spatial subquery is handled by an auxiliary spatial processor, which is part of the extended DBMS, and stores spatial attributes. The query processing strategies are designed to maximize the advantages of using this spatial processor by extending the traditional query handling techniques of decomposition and rewrite rules to work with spatial data. The benefits of GEOQL lie in the simplicity of the extensions, which facilitate the use of existing optimizers.

2.3 GRAL

Becker and Güting [17] approach spatial query processing from a fundamental and formal standpoint. Gral, is an extensible database system with geometric data types and a geometric query language, called geo-relational algebra. Unlike the 9-intersection formalism [4], however, the classification and semantics of the spatial relations between objects are not explicitly specified. In Gral, the query language and the executable query plans are based on a many-sorted algebra. This formalism defines the source and target languages of the optimization, as well as the optimization rules themselves, providing the major advantage of a uniform framework for extensibility and optimization strategies. Optimization is then a process of transformation and translation of algebraic expressions. The rules governing these transformations and translations can be general and independent of the operations in the query language or they can be specialized, facilitating the inclusion of new spatial data types and operations.

2.4 Spatial Joins

Günther [18] examines the efficient computation of spatial joins where two datasets are related to each other by some spatial constraint. Traditional join processing strategies are ill-suited for this problem, because there is no linear ordering that preserves spatial proximity and hence sort-merge cannot be used. Günther describes a class of tree structures called generalization trees that can be used to devise efficient means of join computations. The generalization tree is suited for any application where the data has inherent containment hierarchies. Spatial joins are computed by considering nodes at a higher level in the tree in order to determine the branches that may contain candidates on which the spatial join condition is performed. The concept of hierarchy of detail is applicable to topological relations in the 9-intersection model too, with MBRs forming the coarse level and the actual geometry being the detailed level.

2.5 Comparison with Present Work

Our concern is more fundamental and yet also more abstract than previous approaches in that it addresses issues at a conceptually higher level. The fundamental question being investigated is that of a spatial data model and a formal specification of this model. We use the mathematical definition of binary topological relations [4]. In particular we propose means of reducing the computations involved in evaluating a topological spatial constraint. The strategies presented here have three key benefits: (1) they are based on a well-defined formalism, (2) they are customizable so that users or spatial database administrators may redefine or limit the topological relations needed for their specific application domain and still use the strategies presented here, and (3) the strategies can take into account important statistical information about the data, the probability distribution, or estimates, of the occurrence of various topological relations.

3. 9-Intersection Model

Topological relations are spatial relations that are preserved under such transformations as rotation, scaling, and rubber sheeting. The model for binary topological relations used in this paper is based on the usual concepts of point-set topology with open and closed sets. The binary topological relation between two objects, A and B , in \star is based upon the intersection of A 's interior (A°), boundary (A), and exterior (A^-) with B 's interior (B°), boundary (B), and exterior (B^-). The

nine intersections between the six object parts describe a topological relation and can be concisely represented by a 3×3 -matrix ρ , called the *9-intersection* (Equation 1).

$$\rho(A, B) = \begin{matrix} & A^{\circ} & B^{\circ} & A^{-} & B^{-} & A^{\circ} & B^{-} \\ A^{\circ} & A^{\circ} & B^{\circ} & A^{-} & B^{-} & A^{\circ} & B^{-} \\ A^{-} & B^{\circ} & A^{-} & B^{-} & A^{-} & B^{-} & B^{-} \end{matrix} \quad (1)$$

By considering the values empty (0) and non-empty (1), one can distinguish between $2^9 = 512$ binary topological relations. Only a small subset of them can be realized when the objects of concern are embedded in \mathbb{R}^2 [19]. For two regions (connected, homogeneously 2-dimensional areas with connected boundaries) in \mathbb{R}^2 , there are eight relations which provide a mutually exclusive complete coverage [4] (Figure 1). The terms adopted for them (*disjoint*, *meet*, *overlap*, *equal*, *contains*, *inside*, *covers*, and *coveredBy*) will be often shortened with *d*, *m*, *o*, *e*, *ct*, *i*, *cv*, *cB*.

0 0 1 0 0 1 1 1 1 disjoint	1 1 1 0 0 1 0 0 1 contains	1 0 0 1 0 0 1 1 1 inside	1 0 0 0 1 0 0 0 1 equal
0 0 1 0 1 1 1 1 1 meet	1 1 1 0 1 1 0 0 1 covers	1 0 0 1 1 0 1 1 1 coveredBy	1 1 1 1 1 1 1 1 1 overlap

Figure 1: The eight topological relations between two spatial regions and their corresponding 9-intersection matrices [2].

4. Exploiting Minimal Bounding Rectangles

Geographic databases that employ one of the common spatial access methods [20, 21] usually store for each spatial object its MBR. An MBR is an X-Y-parallel rectangle that fully encloses the geometry of the spatial object. While the MBR is usually only a crude approximation of the object's geometry, it is sufficient in most cases to locate the object in space; therefore, spatial access methods use MBRs as the primary criterion when selecting the physical page on which the spatial object will be stored. Likewise, when accessing objects based upon their spatial location, the MBR is used as a filter to determine whether it is worthwhile to load an object from disk into memory for more detailed tests.

When processing topological queries, the situation is somewhat different. Due to the potential inconsistencies between topology and metric information, no immediate conclusions can be drawn from the relations among the approximations. Occasionally, however, the MBR information can be

exploited to make very fast decisions as to whether an object qualifies to fulfill a particular relation or not. This filter method is based on the fact that there are some consistent mappings between MBR relations and the topological relations of interest. Subsequently, we will develop the mappings between MBR relations and the topological relations between regions. Similar correspondences can be found for the more extensive region-line and line-line relations.

Provided the MBRs are “tight” approximations, i.e., no space is wasted, and no inconsistencies due to the number system exist, the mappings in Table 1 hold from the topological relations between MBRs onto the topological relations between the exact geometric representations. This set of mappings is relevant for queries in which one asks for the topological relation between two given objects. The most dramatic performance improvement will occur if the MBRs are *disjoint*, because this implies that their topological relation is *disjoint* as well. No further analysis of the topological relation will be necessary. Considerable improvements are also obtained if the MBRs *meet*, because in this case only the boundary-boundary intersection has to be evaluated.

MBR relation	topological relation
d_M	d_r
m_M	$d_r \quad m_r$
o_M	$d_r \quad m_r \quad o_r$
i_M	$d_r \quad m_r \quad o_r \quad i_r \quad cB_r$
cB_M	$d_r \quad m_r \quad o_r \quad i_r \quad cB_r$
ct_M	$d_r \quad m_r \quad o_r \quad ct_r \quad cv_r$
cv_M	$d_r \quad m_r \quad o_r \quad ct_r \quad cv_r$
e_M	$e_r \quad o_r \quad cB_r \quad cv_r$

Table 1: Mappings from MBR relations onto exact topological relations.

From Table 1, the reverse mapping from exact topological relations onto MBR relations can be derived in a straightforward manner (Table 2). This mapping indicates in which cases the consideration of MBR relations will improve performance of queries asking for all objects that satisfy a given topological constraint with respect to a given object. The use of the MBR here is as a filter since it eliminates all those cases that can be disregarded right away, because they will never fulfill the particular topological constraint. For example, if the query is to find all objects that are *equal* to a given object A , one first maps the topological relation e_r onto e_M (Table 1) to make a fast evaluation based on the spatial access method. Afterwards, for each object B whose MBR is *equal* to A 's MBR, one has to evaluate whether the topological relation between the exact representations is $e_r \quad o_r \quad cB_r \quad cv_r$ using the reverse mappings (Table 2).

topological relation	MBR relations
d_r	d_M m_M o_M i_M c_{BM} ct_M cv_M
m_r	m_M o_M e_M i_M c_{BM} ct_M cv_M
o_r	o_M e_M i_M c_{BM} ct_M cv_M
i_r	i_M c_{BM}
ct_r	ct_M cv_M
c_{B_r}	i_M c_{BM} e_M
cv_r	ct_M cv_M e_M
e_r	e_M

Table 2: Mappings from exact topological relations onto MBR relations.

5. Queries on Whether Two Objects Satisfy a Set of Topological Relations

The set of intersections to be calculated must be such that the content, empty or non-empty, of the intersections is sufficient to identify the relation R_j . This leads to the definition of a *minimal subset* of the nine intersections. The selection of the subset, which uniquely characterizes a relation R_j , can be optimally determined for each of the relations. For example, the relation *meet* between two regions is characterized by the following 9-intersection matrix:

$$M_{meet} = \begin{matrix} & & 0 & 0 & 1 \\ 0 & 1 & 1 & & \\ 1 & 1 & 1 & & \end{matrix} \quad (2)$$

Comparing this matrix with the matrices of the other 7 relations (Figure 1), it can be determined that only M_{meet} matches the template $\begin{matrix} & & 0 & & 1 \\ 0 & 1 & 1 & & \\ 1 & 1 & 1 & & \end{matrix}$, where each \square indicates a *don't care* value because

only the relation *meet* has a non-empty boundary-boundary intersection *and* an empty interior-interior intersection. Thus the minimal subset for *meet* is {interior-interior, boundary-boundary}.

The following is an algorithm to determine the minimal subset. It is assumed that the computational cost of each of the nine possible intersections, for an arbitrary 9-intersection, is equal. This problem of identifying the minimal distinguishing subset is a form of the Minimum Set Cover problem [22]. We use a greedy heuristic [23] to obtain the minimal sets in our application. R is the set of binary topological relations. A and B are disjoint subsets of R such that $A \cup B = R$. M is the set of 9-intersection matrices for R . Each M_i in M is a 3x3 binary valued matrix. D is the set of XOR matrices D_{ij} where D_{ij} is M_{a_i} XOR M_{b_j} . Thus $D_{ij}[k, l] = 1$ where $k, l \in \{^{\circ}, -, -\}$, if that intersection value distinguishes relation a_i from b_j .

Algorithm 1: Determine the minimal subset, P , of the 9-intersection that distinguishes between mutually disjoint subsets A and B of the set of relations R .

procedure MinimalSet (R, A, B, M)

for each relation a_i in A

for each relation b_j in B

Construct the XOR matrices $D_{ij} = M_{a_i} \oplus M_{b_j}$;

Construct the set of sets I , with elements I_{kl} , where $k, l \in \{^{\circ}, -, -\}$ and I_{kl} is the set of a_i - b_j relation pairs such that $M_{a_i}[k, l] \neq M_{b_j}[k, l]$.

Apply the minimum set cover [21] to find the subset of I such that their union is $\{a_i \times b_j \mid a_i \in A \text{ and } b_j \in B\}$. From this subset create $P_{kl} = \{(k, l) \mid k, l \in \{^{\circ}, \text{ }, -\}\}$. P_{kl} is the minimal set of intersections that is sufficient to distinguish relations in A from those in B .

return P_{kl} ;

For the set of eight region-region relations, the following templates represent the minimal subsets.

$$\begin{array}{cccc}
 R_{\text{contains}} = & \begin{array}{cc} 1 & \\ 0 & \end{array} & R_{\text{inside}} = & \begin{array}{cc} 1 & 0 \\ 0 & \end{array} & R_{\text{disjoint}} = & \begin{array}{cc} 0 & \\ 0 & \end{array} & R_{\text{covers}} = & \begin{array}{cc} 1 & \\ 0 & 1 \end{array} \\
 R_{\text{coveredBy}} = & \begin{array}{cc} 0 & \\ 1 & 1 \end{array} & R_{\text{overlap}} = & \begin{array}{cc} & 1 \\ 1 & \end{array} & R_{\text{equal}} = & \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} & R_{\text{meet}} = & \begin{array}{cc} & 0 \\ 1 & \end{array}
 \end{array}$$

The worst case is for *covers* and *coveredBy*, where three intersection values must be computed, while all other relations can be uniquely determined with only two intersections. This is a considerable improvement over computing all nine intersections.

The same algorithm, used to determine the minimal sets above, applies to queries with disjunctions of topological relations over the same objects. For example, Algorithm 1 used on the constraints in the query, "Find object b such that A contains b \vee A covers b \vee A equal b ," gives the minimal set represented by the following template:

$$R_{ct \vee cv \vee e} = \begin{array}{cc} 1 & \\ 0 & \end{array} \quad (3)$$

Therefore, for each object b , only one intersection (between A 's exterior and b 's interior) must be computed *in lieu* of seven (2 each for *contains* and *equal*, and 3 for *covers*).

Only disjunctions of the base relations are meaningful combinations. Conjunctions of relations over the same objects are not meaningful since each basis set consists of mutually exclusive relations [4], therefore, queries with conjunctions of topological relations over the same objects result in an empty set. Since the set of relations is closed, negations can be expressed as the complement, i.e., as disjunctions of the remaining relations.

6. Queries on the Topological Relation Between Two Objects

Queries on the topological relation between two objects are computationally expensive since the relation between two given objects must be uniquely determined. The basic strategy for such queries is the construction of a decision tree that partitions the search space at each node, thereby progressively excluding all the other relations. In the best case, at each step of the decision process the search space gets partitioned into two halves, therefore requiring $\log n$ computations, where n is the number of relations. In the case of region-region relations, $n = 8$ and so at most three steps are necessary.

The decision tree for topological relations characterized by 9-intersections is based on the values of these intersections and different trees will be obtained by selecting different discriminants at the nodes. For example, if region-region relations are being considered and the discriminant at the root of the tree is the boundary-boundary intersection, then the relations in the \emptyset -valued and $\neg\emptyset$ -valued child nodes are $\{disjoint, contains, inside\}$ and $\{meet, equal, covers, coveredBy, overlap\}$, respectively. Hence for the specific problem at hand the decision tree will not be perfectly balanced. The goal however is to build a near-optimal tree.

6.1 Naive Cost Model

The *naive cost model* assumes that all relations occur with equal probability. For a given tree, T , the computation cost, l_r^T , associated with a relation r is

$$l_r^T = w_r d_r^T \quad (4)$$

where d_r^T is depth of the node at which a particular relation is identified and w_r is the weight associated with relation r .

Definition 1. The *characteristic*, l^T , of a tree T is the sum of the computation costs for all relations, that is,

$$l^T = \sum_r l_r^T = \sum_r w_r d_r^T \quad (5)$$

For the naive model, the characteristic of a tree is the average path length, since $w_r = \frac{1}{N}$ for all N relations. Figure 2 shows the decision tree for the eight region-region that results from using a naive cost model.

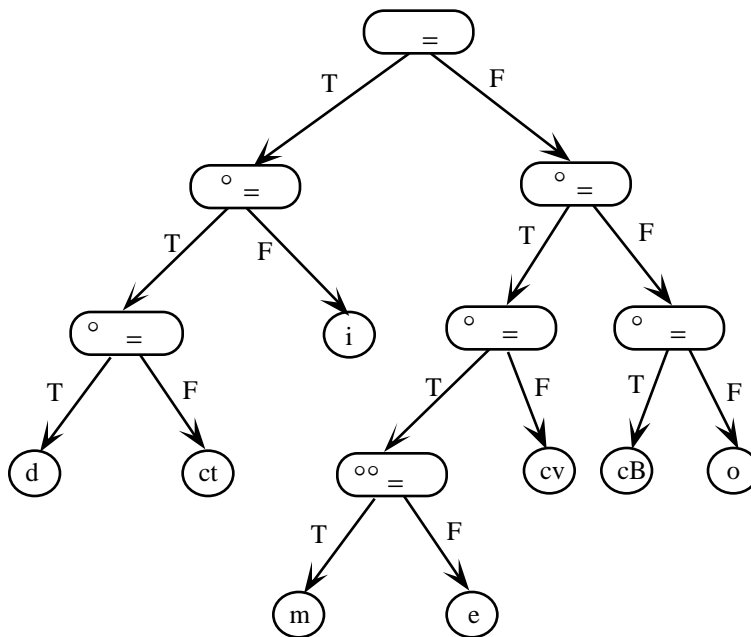


Figure 2: The decision tree for the eight region-region relations based on the naive cost model.

6.2 Refined Cost Model

The naive model can be refined by assuming a certain frequency distribution of relations and a different cost for each operation. A refined model would assign a frequency distribution to the instances of the relations in a database. For example, in a cadastral map of n parcels a fair estimate is that 95% of all relations are *disjoint*, while the remaining 5% are *meet*. The cost of each operation, i.e., computing an intersection, is highly dependent on the choice of data structure and spatial access methods. Operation costs do not influence the choice of the tree, since on average all operations need to be calculated once in the tree; therefore, the characteristic of the tree is stable with respect to the cost of the operations. This section presents an algorithm to build a near optimal decision tree that considers the expected frequency of occurrence of relations for a particular data set.

To illustrate such a refined cost model, we will use the following distribution for region-region relations: If one computes all n^2 topological relations for a given data set, 80% of all relations are

disjoint; 10% *meet*; 5% *overlap*; 2% are *contains* and *inside* each; 0.4% are *covers* and *coveredBy* each, and 0.2% are *equal*. These percentages correspond to the weights $w_d = 0.8$, $w_m = 0.1$, etc.

The following is a greedy algorithm for building a decision tree. At each node if a particular relation is being identified it continues with that one, else the relation with the highest weight is considered. L contains the list of relations while P is a set of intersections (e.g., interior-interior) used so far. Initially $P = \{\}$. When a leaf node is reached, P contains the set of intersections that are sufficient for identifying the relation in the leaf node. M is the set of 9-intersection matrices for the relations in L .

Algorithm 2. Build a decision tree, T , for determining the relation between objects.

procedure BuildTree (L, P, M);

$k := \text{first}(L)$;

Now consider the template R_k ;

Create the list of positions $[x, y]$ such that $R_k[x, y]$

for each position $[x, y]$ that is not in the set P of positions used

for each relation i in list L

if $R_i[x, y]$ **then** increment the count for position $[x, y]$;

Choose the position with the highest count. Break ties arbitrarily.

for each relation i in list L

if $M_i[x, y] = \emptyset$ **then** add relation i to the sublist L_T ;

else add relation i to the sublist L_F ;

Add the chosen position to the set of positions used, i.e., $P_T := P \cup \{[x, y]\}$; $P_F := P \cup \{[x, y]\}$;

if $|L_T| > 1$ **then** $T := \text{BuildTree}(L_T, P_T, M)$;

if $|L_F| > 1$ **then** $T := \text{BuildTree}(L_F, P_F, M)$;

return T ;

The algorithm builds the decision tree recursively using the list L of relations sorted in descending order of their expected frequency of occurrence in the database. The first relation, R_k , in this list is considered and a discriminant, by which L will be partitioned into two sublists, is chosen. Each relation in the list L has a set of intersections, for example boundary-boundary, that can be used as a discriminant. For each such intersection a count is kept of its presence in the sets for the relations in L . Now from the set of intersections for R_k that have not been used as a discriminant, the one with the maximum count is chosen as the current discriminant. List L is then partitioned into two sublists L_F and L_T and the corresponding sets P_F and P_T of intersections used as discriminants are also set up for the next recursive call. Note that the sublists are also sorted in descending order and hence if relation R_k is in one of the lists it will be first element of the list. This ensures that relation R_k is identified before any other relation on the list.

Algorithm 2 assures that the most frequently occurring relation is calculated in the minimum number of steps. The same criteria is applied in each node of the decision tree. For example, when Algorithm 2 is applied to region-region relations, with the frequency distribution discussed at the beginning of this section, the ordered list of relations is $L = (d, m, o, ct, i, cv, cB, e)$. *Disjoint* is the first relation considered and position $[1,1]$ denoting the boundary-boundary intersection is the discriminant. Since $M_d[1,1] = \emptyset$, the sublists are $L_T = (d, ct, i)$ and $L_F = (m, o, cv, cB, e)$. The procedure is invoked recursively until the leaf nodes are reached, resulting in the decision tree shown in Figure 3.

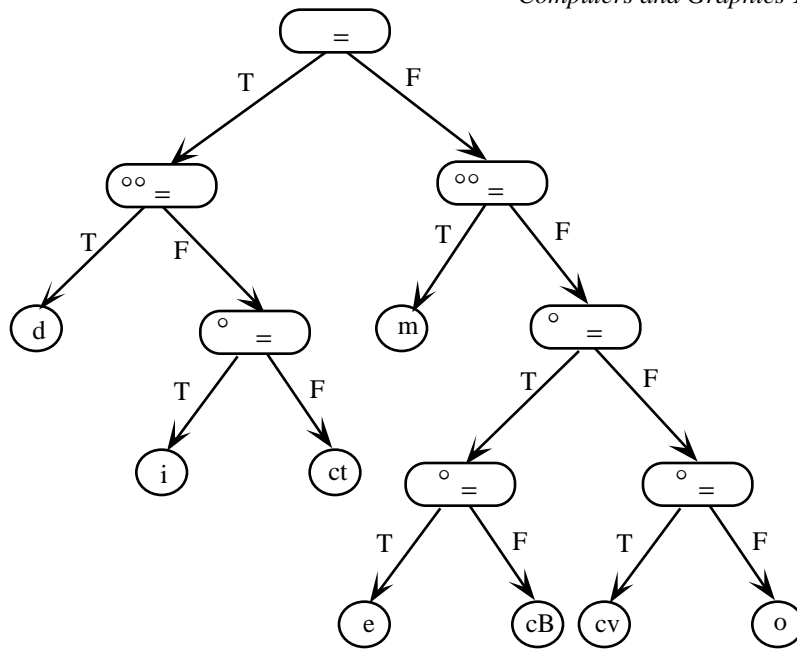


Figure 3: The decision tree for the eight region-region relations based on a refined cost model.

6.3 Characteristic of the Tree

The characteristic of a tree represents the average number of computations that are needed in order to assess a relation. It is defined as the weighted sum of all relations:

$$T = \sum_r w_r d_r^T \quad (6)$$

The tree for the naive cost model (Figure 2) has the characteristic:

$$\begin{aligned} T_n &= 0.8 \times 3 + 0.1 \times 4 + 0.05 \times 3 + 0.02 \times 2 + 0.02 \times 3 + 0.004 \times 3 + 0.004 \times 3 + 0.002 \times 4 \\ &= 3.082 \end{aligned} \quad (7)$$

Using the same distribution, the refined cost model tree (Figure 3)

$$\begin{aligned} T_r &= 0.8 \times 2 + 0.1 \times 2 + 0.02 \times 3 + 0.02 \times 3 + 0.05 \times 4 + 0.004 \times 4 + 0.004 \times 4 + 0.002 \times 4 \\ &= 2.16 \end{aligned} \quad (8)$$

The theoretical lower bound for the characteristic of the decision trees for a given set of relations can be determined by the Huffman algorithm [24]. This bound cannot be achieved because the “codes” —or set of intersection values in this case— that signify a particular relation, are predetermined. The Huffman algorithm, however, determines the optimal prefix codes that should be used. For the distribution considered in this section the lower bound, if the relations were characterizable by Huffman codes, would be 1.422.

The theoretical upper bound is the maximum depth of the decision tree built, assuming a uniform distribution of the relations. For region-region relations this bound would be 4 since *meet* and *equal* are at depth 4 in the tree of Figure 2.

A measure of the optimality for a tree T is given by: $\mu(T) = \frac{T - \text{lower bound}}{\text{upper bound} - \text{lower bound}}$. This has a value 0 for the optimal tree, while it is 1 for the worst tree. For the two characteristics T_r and T_n we have $\mu(T_r) = 0.287$ and $\mu(T_n) = 0.644$; therefore, the tree, T_r , obtained from the refined cost model is closer to the theoretical optimum.

7. Conclusions

We presented a new approach to processing and optimizing queries over spatial relations, which relies on the existence of any of the common spatial access methods, such as R-trees, and exploits the semantics of the relations, heuristics, and the distribution of the objects in geographic space. It builds on the well-defined formalism of the 9-intersection, a model for binary topological relations. This model is popular in GIS research and has been implemented in commercial GISs. We developed algorithms that minimize the calculations of intersections necessary to uniquely determine the topological relation between two given objects, and to find those objects that satisfy a particular topological constraint with respect to a given object. Though the present paper used only the eight region-region relations to demonstrate the principles, the method applies immediately to other topological relations defined by the 9-intersection, such as line-line relations or region-line relations.

The present work assumes that the computation of boundary-boundary intersections has the same cost as say interior-interior intersections. In general, this is not the case. The choice of data model and structure will determine the costs of the operations as will the choice of computational geometry algorithms. For example, using plane-sweep algorithms versus a series of line-line intersection and point-in-polygon tests will enable testing boundary-boundary and interior-interior intersections in the same iteration. Future work will evaluate these effects and tests will be run on datasets such as TIGER files. We also propose to extend the present algorithms to process queries like, "Find all lakes in Maine that are fed by rivers", which have constraints between two regions and between a region and a line.

Acknowledgments

George Markowsky's and Rahul Simha's comments on an earlier version are gratefully acknowledged.

8. References.

1. O. Günther and A. Buchmann. Research Issues in Spatial Databases. *SIGMOD RECORD* **19**(4), 61-68 (1990).
2. M. J. Egenhofer. Reasoning about Binary Topological Relations. in: O. Günther and H.-J. Schek (Ed.), *Advances in Spatial Databases-Second Symposium, SSD '91*. LNCS 525, 143-160, Springer Verlag, Zurich, Switzerland (1991).
3. D. S. Batory, T. Y. Lang, and T. S. Wise. Implementation concepts for an extensible data model and language. *ACM Transactions on Database Systems* **13**(3), 231-262 (1988).
4. M. J. Egenhofer and R. Franzosa. Point-Set Topological Spatial Relations. *International Journal of Geographic Information Systems* **5**(2), 161-174 (1991).
5. M. J. Egenhofer and J. Herring. Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Department of Surveying Engineering, University of Maine, Technical Report (1991).
6. J. R. Herring, R. Larsen, and J. Shivakumar. Extensions to the SQL Language to Support Spatial Analysis in a Topological Data Base. *Proc. GIS/LIS'88*, San Antonio, TX, 741-750 (1988).
7. D. Mark and M. Egenhofer. An Evaluation of the 9-intersection for Region-Line Relations. *Proc. GIS/LIS '92*, San Jose, CA, (1992).
8. M. J. Egenhofer and K. Al-Taha. Reasoning about Gradual Changes of Topological Relationships. in: A. U. Frank, I. Campari, and U. Formentini (Ed.), *Theories and Models of Spatio-Temporal Reasoning in Geographic Space*. LNCS 639, 196-219, Springer Verlag, Pisa, Italy (1992).
9. C. Freksa. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence* **54**, 199-227 (1992).
10. M. J. Egenhofer, A. Frank, and J. Jackson. A Topological Data Model for Spatial Databases. in: A. Buchmann, O. Günther, T. Smith, and Y. Wang (Ed.), *Design and Implementation of Large Spatial Databases*. LNCS 409, 271-286, Springer Verlag, Santa Barbara, CA (1989).
11. J. Herring. The Mathematical Modeling of Spatial and Non-Spatial Information in Geographic Information Systems. in: D. Mark and A. Frank (Ed.), *Cognitive and Linguistic Aspects of Geographic Space*. 313-350, Kluwer Academic, Dordrecht (1991).
12. M. J. Egenhofer. Why not SQL! *International Journal of Geographical Information Systems* **6**(2), 71-85 (1992).
13. M. J. Egenhofer, E. Clementini, and P. Di Felice. Topological Relations Between Regions with Holes. *International Journal of Geographical Information Systems* **8**(2), (1994).
14. W. G. Aref and H. Samet. Optimization Strategies for Spatial Query Processing. *Proc. 17th International Conference on Very Large Databases*, Barcelona, Spain, 81-90 (1991).

- Computers and Graphics* 18 (6): 815-822, 1994.
15. W. G. Aref and H. Samet. Extending a DBMS with Spatial Operations. in: O. Günther and H.-J. Schek (Ed.), *Advances in Spatial Databases—Second Symposium, SSD'91*. LNCS 525, 299-318, Springer-Verlag, Zurich, Switzerland (1991).
 16. B.-C. Ooi and R. Sacks-Davis. Query Optimization in an Extended DBMS. in: W. Litwin and H.-J. Schek (Ed.), *Foundations of Data Organization and Algorithms*. LNCS 367, 48-63, Springer Verlag, Paris, France (1989).
 17. L. Becker and R. H. Güting. Rule-Based Optimization and Query Processing in an extensible Geometric Database System. *ACM Transactions on Database Systems* 17(2), 247-303 (1992).
 18. O. Günther. Efficient Computation of Spatial Joins. *Proc. Ninth International Conference on Data Engineering*, Vienna, Austria, 81-90 (1993).
 19. M. J. Egenhofer and J. Herring. A Mathematical Framework for the Definition of Topological Relationships. *Proc. Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, 803-813 (1990).
 20. H.-P. Kriegel, M. Schiwietz, R. Schneider, and B. Seeger. Performance Comparison of Point and Spatial Access Methods. in: A. Buchmann, O. Günther, T. Smith, and Y. Wang (Ed.), *Design and Implementation of Large Spatial Databases*. LNCS 409, 89-114, Springer-Verlag, Santa Barbara, CA (1989).
 21. H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, Reading, MA (1989).
 22. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA (1979).
 23. V. Chavtal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research* 4(3), 233-235 (1979).
 24. D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 40(9), 1098-1101 (1952).