Query Pre-Processing of Topological Constraints: Comparing a Composition-Based with Neighborhood-Based Approach*

M. Andrea Rodríguez[1], Max J. Egenhofer[23], and Andreas D. Blaser[4]

[1]Department of Information Engineering and Computer Science, University of Concepción
Edmundo Larenas 215, Concepción, Chile.
[2]National Center for Geographic Information and Analysis, University of Maine,
Orono, ME 04469-5711, USA
[3]Department of Information Science, University of Maine,
Orono, ME 04469-5711, USA
[4] Environmental Systems Research Institute,
380 New York Street, Redlands, CA 92373-8100, USA
andrea@udec.cl, max@spatial.maine.edu,abl@esri.com

Abstract. This paper derives and compares two strategies for minimizing topological constraints in a query expressed by a visual example: (1) elimination of topological relations that are implied uniquely by composition and (2) restriction to topological relations that relate near-neighbor objects, as determined by a Delaunay triangulation. In both cases, the query processing approach is to solve a constraint satisfaction problem over a graph of binary topological relations. Individuals and the combination of the composition- and neighborhood-based strategies were implemented and compared with respect to their ability to reduce topological constraints, and with respect to the quality of the results obtained by a similarity-based searching that uses these pre-processing strategies. The main conclusion of this work is that similarity queries that are formulated in a visual language should exploit the metric characteristics of the configuration, even if only topological constraints are considered for making matches.

1 Introduction

Query evaluation in geographic databases is often expensive, because the spatial data stored are more complexly structured and data sets are larger than their non-spatial counter parts. Spatial queries are usually expressed as a set of spatial objects and a set of spatial constraints among the objects. The spatial constraints may be topological

* This work was partially funded by FONDECYT 1010897 and 7010897 and the National Imagery and Mapping Agency under grant number NMA202-97-1-1023. Max Egenhofer's work is further support by grants from the National Science Foundation under grant numbers IIS-9970123 and EPS- 9983432, the National Imagery and Mapping Agency under grant numbers NMA201-01-1-2003, NMA201-00-1-2009, NMA401-02-1-2009, and National Institute of Environmental Health Sciences, NIH, under grant number 1 R 01 ES09816-01.

(e.g., *overlap* or *inside*), metric (e.g., *within 2 miles* or *near*), and directional (e.g., *north*). The goal of a query evaluation is to match the query constraints among the related objects with binary spatial relations between objects that are stored in a spatial database. Since the spatial objects are embedded in the same space, the set of binary spatial relations—and, therefore, also the number of possible constraints—grows exponentially with the number of spatial objects in the query. This is the case, for example, if the goal is to find spatial scenes that are similar to a given configuration or if a spatial query is derived from a sketch [1], where the user specifies a query by drawing an example that is composed of objects and implicit spatial constraints. For n objects drawn, the sketch—and, therefore, the derived query—contains n^2 topological relations, n^2 direction relations, and n^2 metric relations. Such a type of query implies very hard combinatorial computations [2-4] and, therefore, presents a challenging problem for spatial information retrieval.

The evaluation of a spatial query is typically composed of a sequence of steps (Figure 1) [5]. Starting with the semantic analysis, a *consistency checker* evaluates whether or not query constraints contain self-contradictions [6]. The *optimizer* aims at speeding up the query processing by generating an evaluation plan according to optimization rules and access path selections. Finally, the *query processor* is in charge of carrying out the evaluation. This paper focuses on spatial query optimization involving binary topological relations.

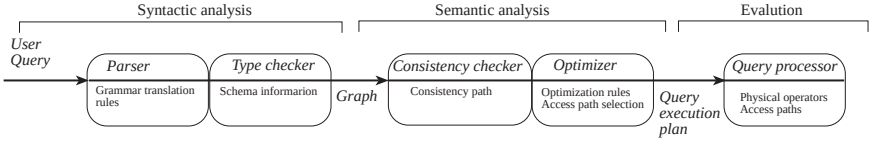


Fig. 1. Sequence of tasks in a query evaluation (based on [5]).

Most approaches to spatial query optimization have pursued an efficient search process by defining an optimal evaluation plan [7-11]. In these approaches, the complexity of spatial operators and the size of the search space become the basis for defining the best sequence of tasks in answering a query. This paper pursues a complementary approach that is independent of the data collection in the spatial database and, therefore, can be performed as part of query optimization in a pre-processing step. Query pre-processor aims at reducing the number of spatial constraints by analyzing the content of a spatial query with the goal to find a subset of constraints that will satisfy the query. A subset of constraints does not lose information if the results of the query process obtained with this subset satisfy the constraints that were not evaluated.

This study considers two strategies for reducing the number of topological relations that describe a spatial configuration: (1) composition-based and (2) neighborhood-based approaches. The composition-based approach considers the spatial reasoning concept of composition, where topological relations can be derived from a subset of given topological relations [12]. It is, therefore, based purely on the algebraic properties of the set of topological relations. The neighborhood-based strategy considers semantics of the space by emphasizing that non-disjoint topological

relations are more relevant than disjoint relations, since they indicate physical connection between objects [13, 14]. Thus, what matters are the relations between objects that are physically close to each other. Although some level of composition-based or neighborhood-based query pre-processing has been used in the past [13, 15, 16], no study has described the background, and the advantages and disadvantages of pre-processing the query with such strategies.

The organization of the remainder of this paper is as follows: Section 2 presents the representation of spatial configurations and definitions associated with topological relations. Section 3 and 4 develop composition-based and neighborhood-based pre-processing strategies, respectively. Experimental results that compare both pre-processing strategies are given in Section 5. Section 6 draws conclusions and discusses future research directions.

2 Representing Topological Relations

A configuration C is a set of objects O and a set of constraints R expressed by the binary topological relations between two objects (Equation 1). This configuration can be seen as a graph, where nodes are the objects and directed edges are the binary topological relations [15, 18].

$$C = \left\{(O, R) : O = \{o_1 ... o_n\} \wedge R = \{(o_i, o_j) : o_i, o_j \in O\}\right\} \tag{1}$$

The graph g that describes a spatial configuration with n objects can be represented as a matrix of nxn elements, where these elements identify binary topological relations R_{ij}. Elements along the matrix diagonal $R_{i,i}$ are all *equal* relations.

2.1 Topological Relations

Topological relations are binary spatial relations that are preserved under topological transformations such as rotation, translation, and scaling. This work concentrates on topological relations between regions. Figure 2 shows the eight topological relations that can be found between two regions [19, 20], organized in a graph that connects conceptual neighbors derived from the concept of *gradual changes* [21]. For example, *disjoint* and *meet* are two neighboring relations in this graph and, therefore, they are conceptually closer than *disjoint* and *overlap*. Only regions are considered here, because objects are usually indexed based on their Minimum Bounding Rectangles (MBRs) [10, 22]. Searching for MBRs is a first filter in solving a query and is usually sufficient for finding a spatial object.

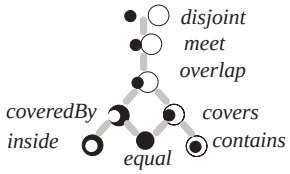


Fig. 2. Eight topological relations between regions, arranged by their conceptual neighborhoods [12].

Refinements of these topological relations can be also introduced in order to differentiate relations by taking into account metric characteristics of objects, such as relative size and distances [23, 24]. Thus, a pair of objects can be seen as further disjoint than another pair if the distance between the first pair is larger than the distance between the second pair.

2.2 Relation Algebra

Topological relations are usually defined as a *relation algebra* [25]. A *relation algebra* (with universe $\Re$) is defined as a ten-tuple $\langle \Re, +, \bullet, -, 0, 1, ;, 1', 0', \overline{\ } \rangle$, where $\langle \Re, +, \bullet, -, 0, 1 \rangle$ is a Boolean algebra, 0 is the *empty* relation, 1 is the *universal* relation, ; is a binary operation called *composition*, 1' is the *identity* relation, 0' is the *diversity* relation, and $\overline{\ }$ is a unary operator forming the *converse* of a given relation [25]. The composition operation (;) allows us to make inferences about the relation between two objects, o_i and o_k, by combining the relations, *R* and *S*, over a common object, o_j (Equation 2).

$$R ; S \equiv (o_i, o_k) \mid \exists\, o_j \text{ such that } (o_i, o_j) \in R \text{ and } (o_j, o_k) \in S \tag{2}$$

2.3 Composition of Topological Relations

The composition may result in a set of relations that is composed of one or more than one element and whose number of elements increases as less precise information is obtained from the inference. For example, if the composition of two operations yields the set with all possible relations (i.e., the universal relation 1), no information at all is obtained from this inference. The composition table for topological relations between regions (Figure 3) shows that out of the 64 compositions, 27 compositions have a unique result, three compositions yield the universal relation, and 34 compositions have between two and six relations in their results.

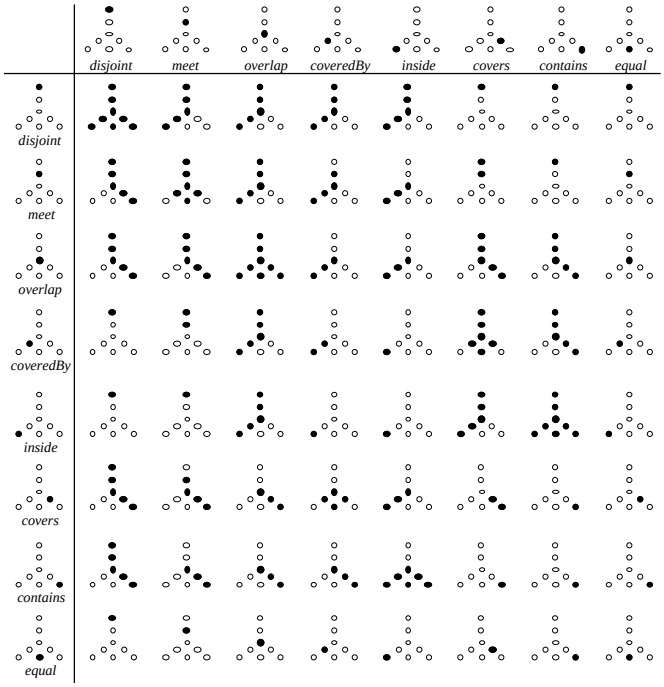


Fig. 3. Composition able of topological relations between regions [12].

3 Composition-Based Pre-Processing

The composition-based approach for query pre-processing is solely based on the algebraic properties of composition. It does not require or exploit any metric descriptions of the objects in a configuration. The composition-based approach starts with a consistent graph and finds a smallest subgraph, from which one can derive the complete, original graph. The strategy for finding this subgraph follows the principles of topological consistency in a graph [26]. There is a comprehensive method for analyzing the consistency of spatial configuration based on the logical consistency expressed by the composition of relations. Given a configuration expressed as a graph, topological consistency is formulated as a constraint satisfaction [16,17] problem over a network of binary topological relations [18, 27]:

- Each node must have a self-loop, denoting the identity relation (Equation 3).

$$\forall_i r_{ii}=1' \tag{3}$$

- For each directed edge from *N* to *M*, there must be an edge in the reverse direction, denoting the converse topological relation (Equation 4).

$$\forall_{i,j}\, r_{i,j} = \overline{r_{j,i}} \tag{4}$$

- Although a variety of paths can lead from one node to another, in order to infer the path consistency of a relation it is sufficient to consider all compositions of path length 2 that connect the relation's two nodes [26]. Having a consistent graph, a topological relation must coincide with its induced relation determined by the intersection of all possible composition path of length 2 (Equation 5).

$$\forall_{i,j}\, r_{i,j} = \bigcap_{k=A}^{N} r_{i,k} ; r_{k,j} \tag{5}$$

Following the principle of path consistency, a relation could be completely derived if and only if it is the unique possible relation that results from the intersection of different path compositions in the query graph.

Having an initial graph that is consistent and complete, we must prove that the minimum subgraph is unique. Otherwise, the algorithm that determines the subgraph would need to choose between different paths. The analysis of the uniqueness of this minimum subgraph is done exhaustively using an algorithm that checks possible composition-based derivations.

Consider each single composition with a crisp result, that is, all compositions whose results have a single relation. To obtain a unique subgraph with three objects, no permutation of the relations of compositions with crisp results should produce another crisp result.

Antecedent: $r_{i,j} = r_{i,k} ; r_{k,j}$, where "=" implies a crisp result.

Hypothesis: $(r_{i,k} \neq r_{i,j} ; r_{j,k}) \wedge (r_{k,j} \neq r_{k,i} ; r_{i,j})$.

Without considering the fifteen trivial compositions with *equal*, there exist twelve crisp results of the topological compositions (Table 1). By checking exhaustively all twelve compositions, we accept the hypothesis. Thus, for single compositions, the derived relation is the only relation that can be derived from the combinations of the three relations involved in the composition (i.e., *R*, *S*, and *T*). An important observation is that *overlap* is the only relation whose participation in a composition operation does not result in a non-trivial crisp result.

Table 1. Crisp results for single composition.

Crisp Result	 *disjoint*	 *inside*	 *contains*
Number of Occurrences	6	3	3

For path consistency it is known that a relation can be derived if the intersection of all possible composition paths of length 2 results in this unique relation (Equation 5). The foundation for this assessment is the set of permutations that can be created by exchanging the derived relation with any of the two composition components in all

possible composition paths of length 2. To check the uniqueness of the minimum graph then requires the analysis of whether or not the intersections of all permutations result in a crisp relation.

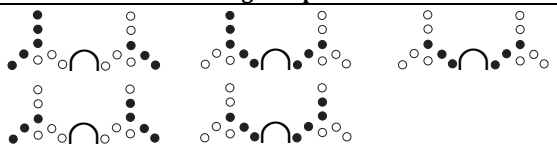
Antecedent: $r_{i,j} = \bigcap_{\forall k, k \neq i \wedge k \neq j} r_{i,k}; r_{k,j}$.

Hypothesis: $\left(r_{i,k} \neq \bigcap_{\forall l, l \neq i \wedge l \neq k} r_{i,l}; r_{l,k} \right) \wedge \left(r_{k,j} \neq \bigcap_{\forall l, l \neq j \wedge l \neq k} r_{k,l}; r_{l,j} \right)$

Using an exhaustive approach we consider, in a first instance, intersections of two paths of length 2. For example, given that $r_{i,j}$ follows from the intersection of $r_{i,k};r_{k,j} \cap r_{i,l};r_{l,j}$, the approach is to check whether or not $r_{i,k} = r_{i,j};r_{j,k} \cap r_{i,j};(r_{l,k};r_{j,k})$.

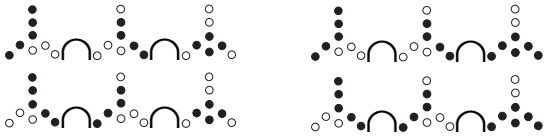
In this process we considered all possible pairs of compositions that do not involve the $equal$ relation (i.e., 49 different compositions). There are 1,176 possible combinations, created by the combination of two compositions over the set of 49 possible compositions. Ninety-eight different intersections create crisp results and do not include a single composition with crisp result. Among these 98 crisp intersections, only ten combinations derive a relation other than *overlap* (Table 2). The permutation of the derived relation by any of the components of the combination proves to give no crisp result. Thus, for configurations with four elements there is just one unique minimum graph, since no exchangeable relation could be derived with the same set of objects.

Table 2. Crisp results for intersections of two compositions with path length 2.

Intersections of Two Compositions with Path Length 2 Yielding Crisp Results	Crisp Result	Number of Occurrences
	 *overlap*	88
	 *covers*	4
	 *coveredBy*	4
	 *meet*	2

Subsequently, crisp results could be derived from the intersection of three paths of length 2. In such a case, there are 162 intersections of three paths with crisp results. This number does not include double paths of compositions or single compositions with crisp results (Table 3). All these intersections produce the relation *overlap*. An exhaustive analysis shows that for configurations with five objects there exists only one unique minimum graph.

Table 3. Crisp results for the 162 intersections of three compositions with path length 3.

Intersections of Three Compositions with Path Length 2 Yielding Crisp Results	Number of Occurrences
	128
	32
	2

Subsequently, the analysis found no further combinations of composition paths whose intersection would produce a crisp result. Indeed, after the intersection of two compositions of length 2, no crisp results other than *overlap* were found. This *overlap* relation does not produce any crisp result when it is composed with other topological relation. So, if we have a configuration with a given set of spatial objects, where one topological relation exists that is completely derivable from the intersection of all possible path of length 2 (Equation 5), we have proved that no permutations of relations that participate in these intersections produce another crisp result. Consequently, no other relation within this configuration can be derivable, and the minimum graph that represents the configuration is unique.

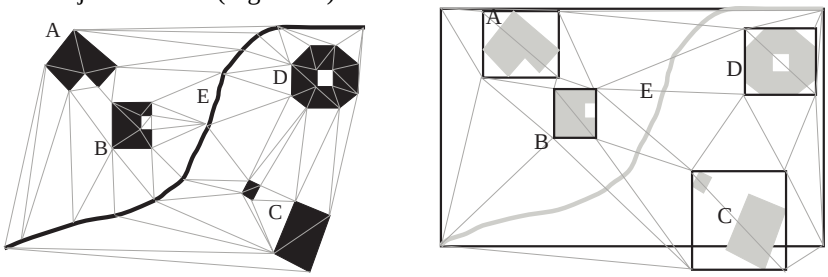
4 Neighborhood-Based Pre-Processing

Pre-processing techniques that concentrate on closely related spatial objects follow Tobler's *First Law of Geography*: "Everything is related to everything else, but nearby things are more related than distant things" [29]. Neighborhood-based pre-processing keeps only the relations in a query graph that represent physically connected objects and near-disjoint relations, but eliminates medium- and far-disjoint. An object is physically connected to another objects if their boundaries are neighbors. There exist different algorithms to establish the neighborhood of spatial objects. Some graph structures that consider the spatial distribution of objects are the Minimum Spanning Tree and the Relative Neighborhood Graph [30, 31].

One of the most widely used methods to connect points in the space is the *Delaunay Triangulation* [32]. It partitions the Euclidean space, composed of a set of points, into triangles such that no four points of this set are co-circular. The dual of the Delaunay Triangulation, the *Voronoi Diagram*, represents a partition of space into regions where points of the Delaunay Triangulation are the nuclei of specific areas. These areas are bounded by the perpendicular bisectors of the nucleus and the set of its neighboring points [33].

A Delaunay Triangulation is based on a set of points. Spatial configurations, however, may be composed of points, lines, or regions. To qualify as neighbors, the

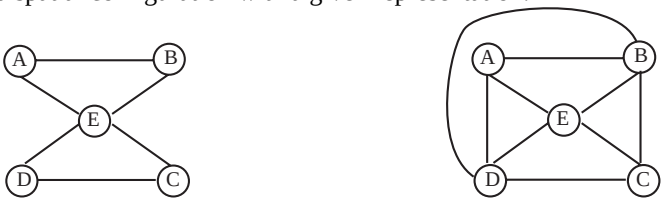
boundaries of two objects must share one or more Voronoi edges. Because of the dual characteristics of the Voronoi and Delaunay Triangulation, a shared Voronoi edge is the same as one or more connecting edges in the Delaunay Triangulation [13]. Figure 4 illustrates two constrained Delaunay Triangulations. They capture the spatial neighborhood at different levels of detail using either the objects' boundaries (Figure 4a) or the objects' MBRs (Figure 4b).



(a) (b)

Fig. 4. Delaunay triangulation of a spatial configuration represented by (a) its objects' edges and (b) its objects' MBRs.

Since the detail in the boundary representations affects the space partition, the Delaunay Triangulation may change depending on the geometric representation of objects. This effect is even clearer as objects are represented by their MBRs using two or four extreme points. This work addresses query pre-processing using the objects' MBRs that are defined by four extreme points. Based on these triangulations we obtained two different subgraphs (Figure 5), where the graph determined based on objects' boundaries represents a subgraph of the graph determined based on objects' MBR. Consequently, although from a theoretical point of view there is a unique minimum subgraph derived from the Delaunay Triangulation, this subgraph may not be the one that is obtained with the Delaunay Triangulation implemented. The implementation, however, is deterministic in the sense that it always finds the same subgraph for a spatial configuration with a given representation.



(a) (b)

Fig. 5. Subgraphs obtained with Delaunay triangulation of a spatial configuration represented by (a) its objects' edges and (b) its objects' MBRs.

Euler's Equation (Equation 6) [33] can be applied for every convex polyhedron with m_v nodes (vertices), m_f faces, and m_e edges. Each edge in a Delaunay Triangulation bounds exactly two vertices. Therefore, if all vertices of a Delaunay Triangulation were substituted with objects and all edges with binary relations, we

can deduce that for a *very large graph* the average number of neighbors (*av_ng*) of an object is less than six (Equation 7). Since in this work four points represented the geometry of an object, and in the extreme case all four points connect to different objects, the number of neighbors of an objects in a *very large graph* is less than 24, that is, it grows linearly by $O(n)$, with n being the number of objects. This upper bound of the number of relations in the final subgraph contrasts the theoretical bound of $O(n^2)$ of a query without pre-processing.

$$m_e \leq 3 \cdot m_v - 6 \tag{6}$$

$$av\_ng = \frac{2m_e}{m_v} \leq 6 - \frac{12}{m_v} \tag{7}$$

5 Experimental Comparison

In this experiment, queries are sketches that provide topological relations as well as metric characteristics of objects. The experiment involves the implementation of a searching mechanism that is based on a content measure. We also used an experimental database that could include regular and extreme cases of realizable configurations, to test whether such cases have an impact on the composition-based or the neighborhood-based pre-processing techniques.

5.1 Experimental Setting

Our analysis applied three different strategies for pre-processing: (1) pre-processing using topological composition $S(g)$, (2) pre-processing using Delaunay triangulation $V(g)$, and (3) pre-processing using Delaunay triangulation followed by composition-based pre-processing ($S(V(g))$).

While composition-based $S(g)$ pre-processing acts over relations (i.e., constraints), neighborhood-based pre-processing $V(g)$ acts over spatial objects. Thus, while eliminating a constraint affects subsequent eliminations when using composition-based pre-processing, eliminating a constraint does not affect the subsequent elimination by the neighborhood-based strategy. This work eliminates only constraints, so $V(S(\ldots()))$ will not differ from $V(g)$. On the other hand, although it is known that $S(S(\ldots S(g)\ldots)$ is the same as $S(g)$, it is impossible to eliminate *a priori* $S(V())$.

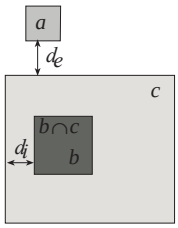
This study evaluates the quality of the pre-processing techniques by using a searching process. This evaluation analyzes whether or not by eliminating constraints through the different pre-processing strategies, the query process can still give good results. The experiment evaluates queries using a *forward checking algorithm* [34] and a *content-based* similarity evaluation of configurations [4]. The content of topological relations in a configurations is defined by a quantitative content measure (Equation 8), which distinguishes the eight topological relations between regions in a plane and introduces metric refinements to differentiate among sizes and distances

between objects [24]. This content measure is independent of rotation and horizontal or vertical flipping of configurations.

$$F_m(A, B) = \frac{area(A) - 2area(A \cap B)}{area(A)} + \frac{distance(\delta A, \delta B)}{diagonal(A)}, \quad (8)$$

$$F_m(B, A) = \frac{area(B) - 2area(A \cap B)}{area(B)} + \frac{distance(\delta B, \delta A)}{diagonal(B)} \quad \text{where}$$

$$distance(\delta A, \delta B) = \begin{cases} d_e(\delta A, \delta B) \text{ if } A \cap B = \varnothing \\ -d_i(\delta A, \delta B) \text{ if } A \cap B \neq \varnothing \end{cases}$$



We define a similarity value between configurations as the inverse of the distances between content-measure values of pair of objects in the first and second configuration (Equation 9).

$$D(Q, S) = \sum_{v_i, v_j \in Q, u_i, u_j \in S} \sqrt{\left(F_m(v_i, v_j) - F_m(u_i, u_j)\right)^2 + \left(F_m(v_j, v_i) - F_m(u_j, u_i)\right)^2} \quad (9)$$

The forward checking strategy takes the constraints one-by-one and searches for pairs of objects that satisfy this constraint [22]. Then, it performs a *join operation* to combine the results of the search of objects that satisfy individual constraints. A constraint is considered satisfied if the difference between the content measure of the objects in a query and the content measure of the objects in the solution is less than or equal to a threshold (0.01 in this case).

5.2 Data Domain

To perform the experiment we created a database of 2,025 elements from all possible objects that fit in a 9x9 box, considering objects whose edge lengths vary from 1 to 9. From this database of 2,025 objects we created a domain of topological relations with a total of 758,614 *disjoint*, 192,464 *meet*, 851,328 *overlap*, 25,200 *coveredBy*, 2,024 *inside*, 173,620 *covers*, and 44,100 *contains* relations. In order to speed up the searching process, binary relations, which are defined as tuples of the content measure ($F(o_i,o_j)$,$F(o_j,o_i)$), were indexed using an R-Tree-like structure [4]. Unlike the traditional use of the R-Tree structure, which organizes objects by their physical locations, the R-tree structure is used here for organizing content values of topological relations [4]. In this experiment, 98 queries with five objects were randomly created, 52 configurations when the pre-processing based on composition reduces the number of constraints, and 46 configurations when pre-processing based on composition does not change the initial configurations.

Table 4. Topological queries and their pre-processing results with relations *disjoint* (d), *meet* (m), *overlap* (o), *inside* (i), *contains* (c), *covers* (cv), *coveredBy* (cb), and *equal* (e).

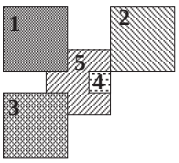
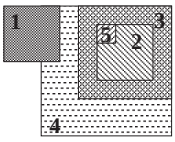
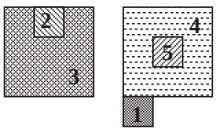
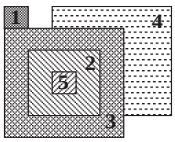
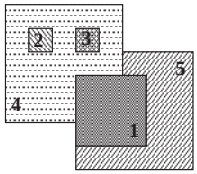
G	Query	*S*(*g*)	*V*(*g*)	*S*(*V*(*g*))
a		- d d d o - - d m m - - - d o - - - - cb - - - - -	- d d d o - - - m m - - - d o - - - - cb - - - - -	- d d d o - - - m m - - - d o - - - - cb - - - - -
		S(a) ⊃ *S*(*V*(a))	*V*(a) ⊂ *S*(a)	*S*(*V*(a)) = *V*(a)
b		- d d d d - - d d d - - - d o - - - - d - - - - -	- d - - - - - d - - - - - d - - - - - d - - - - -	- d - - - - - d - - - - - d - - - - - d - - - - -
		S(b) ⊃ *S*(*V*(b))	*V*(b) ⊂ *S*(b)	*S*(*V*(b)) = *V*(b)
c		- - d o - - - i - cv - - - cb - - - - - - - - - - -	- d d o d - - i i cv - - - cb c - - - - c - - - - -	- - d o - - - i - cv - - - cb - - - - - - - - - - -
		S(c) = *S*(*V*(c))	*V*(c) ⊃ *S*(c)	*S*(*V*(b)) ⊂ *V*(c))
d		- - d m - - - cb - - - - - d - - - - - c - - - - -	- - d m d - - cb - - - - - - - - - - - c - - - - -	- - d m - - - cb - - - - - - - - - - - c - - - - -
		S(d) ⊃*S*(*V*(d))	*V*(d) ≠ *S*(d)	*S*(*V*(d)) ⊂ *V*(d))
e		- - d m - - - i - c - - - - - - - - - cv - - - - -	- d m - - - - i o c - - - o - - - - - cv - - - - -	- - d - - - - i - c - - - o - - - - - cv - - - - -
		S(e) ≠ *S*(*V*(e))	*V*(e) ≠ *S*(e)	*S*(*V*(e)) ⊂ *V*(e)
f		- - d o cb - - d i d - - - i m - - - - - - - - - -	- d d o cb - - d i - - - - i - - - - - o - - - - -	- d d o cb - - d i - - - - i - - - - - o - - - - -
		S(f) ≠ *S*(*V*(f))	*V*(f) ≠ *S*(f)	*S*(*V*(f)) = *V*(f)

Table 4 presents six of the total of queries (g) analyzed by using $S(g)$, $V(g)$, and $S(V(g))$ pre-processing. Queries represent different combinations of results after applying the pre-processing strategies. The first two queries do not have any change after $S(g)$ pre-processing, while they have one or six less relations after $V(g)$ pre-processing. In particular, the second query is an extreme case by using neighborhood-based preprocessing with six eliminations. The opposite situation occurs for the third query, where $V(g)$ composition has no changes and $S(g)$ pre-processing eliminates five relations. The last tree queries have fewer relations after both $S(g)$ and $V(g)$ pre-processing; however, they differ in how S(g), V(g), and $S(V$(g)) are related.

5.3 Results

With respect to the sets of constraints obtained after the query pre-processing, the experiment shows that $S(V(a))$ can reduce the number of edges more than S(a) and V(a) do (Query *d*). For all three strategies, the maximum number of eliminated edges in a query with five objects was six. For all pre-processing strategies, *converse* and *identity* operators over topological relations were applied, such that the experiment considers topological relations only in one direction (upper half of the matrix that represents the graph). Six association rules were derived from the analysis of the results (Equations 10-15).

$$(S(g) = g) \rightarrow (V(g) = S(V(g))) \tag{10}$$

$$(V(g) = g) \rightarrow (S(g) = S(V(g))) \tag{11}$$

$$(S(g) \subset V(g)) \rightarrow ((S(g) = S(V(g))) \wedge (S(V(g)) \subset V(a))) \tag{12}$$

$$(S(g) = V(g)) \rightarrow ((V(g) = S(V(g))) \wedge (S(g) = S(V(g)))) \tag{13}$$

$$(V(g) \subset S(g)) \rightarrow ((V(g) = S(V(g))) \wedge (S(V(g)) \subset S(g))) \tag{14}$$

$$(V(g) \neq S(g)) \rightarrow \neg(S(g) = S(V(g))) \tag{15}$$

With respect to the results of the search process, for all six queries the algorithm with or without query pre-processing finds the right solutions, that is, configurations that are equal to the queries (i.e., optimal solutions). Since the content measure considers the relative size and position of objects, while it disregards differences due to rotation and flipping, the algorithm finds more than one optimal solution. These additional optimal solutions are equivalent to the query if configurations are rotated or flipped over the horizontal or vertical axis.

Results of the search process vary among queries (Table 5). The number of solutions in each query increases as pre-processing eliminates constraints. This increment depends not only on the number of constraints eliminated, but on the type of these constraints. *Disjoint* and *overlap* relations are far more frequent and, therefore, constraints based on these relations will have more candidate solutions. In terms of the satisfaction of topological constraints, while composition-based pre-processing guarantees that topological constraints are satisfied, neighborhood-based cannot guarantee this satisfaction. In the worst case, neighborhood-based pre-processing obtains solutions where 4 of the constraints were not satisfied and, in the worst average, 2 constraints where not satisfied. It is important to note that although

neighborhood-based pre-processing may loose information of the query, the algorithm will always find the optimal solutions, if they exist, and the ranking of the solutions will place optimal solutions first.

Table 5. Search results in terms of number and similarity values (distance 0 stands for an optimal solution).

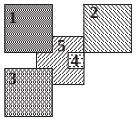
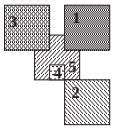
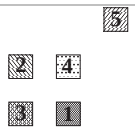
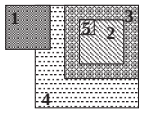
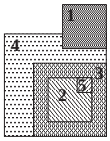
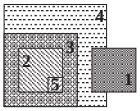
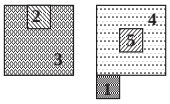
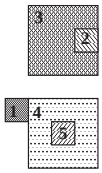
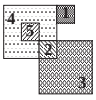
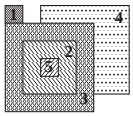
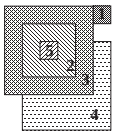
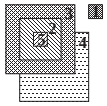
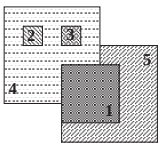
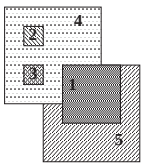
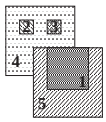
	Query	*G*	*S*(*g*)	*V*(*g*)	*S*(*V*(*g*))
a	Solutions	45	45	45	45
	Solutions with constraint violations	0	0	0	0
	Maximum constraint violations	0	0	0	0
	Average Distance	0	0	0	0
	CPU time [seconds]	399.24	399.24	406.57	406.57
b	Solutions	876	876	45,648	45,648
	Solutions with constraint violations	0	0	0	0
	Maximum constraint violations	0	0	0	0
	Average Distance	0	0	1.45	1.45
	CPU time [seconds]	25.35	25.35	4.66	4.66
c	Solutions	24	720	24	720
	Solutions with constraint violations	0	0	0	0
	Maximum constraint violations	0	0	0	0
	Average Distance	0	0.08	0	0.08
	CPU time [seconds]	0.16	0.08	0.16	0.08
d	Solutions	5	1,024	982	65,996
	Solutions with constraint violations	0	0	790	501.88
	Maximum constraint violation	0	0	4	4
	Average Distance	0	0.32	0.96	1.13
	CPU time [seconds]	594.22	591.71	596.61	587.49
e	Solutions	94	244	450	594
	Solutions with constraint violations	0	0	300	416
	Maximum constraint violations	0	0	1	1
	Average Distance	0	0.06	0.24	0.27
	CPU time [seconds]	0.44	0.76	0.11	0.01
f	Solutions	48	96	96	96
	Solutions with constraint violations	0	0	48	48
	Maximum constraint violations	0	0	1	1
	Average Distance	0	0.001	0.07	0.07
	CPU time [seconds]	19.72	79.03	8.78	8.78

In terms of performance, there is no clear relationship between time and number of constraints. Since queries are of small number of objects, time pre-processing does not affect the overall time. Although in most cases, neighborhood-based pre-processing tends to reduce the CPU time of query evaluation, composition-based pre-processing is not always efficient in doing so. In all cases CPU time is strongly influenced by the number of constraints in the query and the frequency of relations in the database. As the number of candidate relations in the database that correspond to a query constraint increases, the computational cost grows, since more candidate objects have to be combined. In addition, less constrained queries may also increase the CPU Time. This is the case, for example, of Query *a*, where the CPU time for the

original query was less than the reduced query ($V(a)$), since an elimination of a constraint (i.e., *disjoint*) may increases the domain of search.

The metric refinements make further distinctions among topological relations that allow us to reduce the time needed to find a candidate solution. For example, since *disjoint* relations are more frequent in the database, one could think that queries based on this relation will need more processing time. The metric refinements of topological constraints, however, differentiate among types of *disjoint* relations, reducing candidate solutions of query constraints and, therefore, reducing the processing time (e.g., Query *b*).

Table 6. Best and worst results of six test queries

	Query	Result with Best Match	Worst Result
a			
b			
c			
d			
e			
f			

A possible explanation for the better performance with the neighborhood-based pre-processing—when both strategies eliminate constraints—is that objects may have many *disjoint* relations; however, *closely disjoint* objects are less frequent than *far apart* objects. Neighborhood-based pre-processing will always keep *closely disjoint* objects, while it eliminates *far apart* objects. Composition-based pre-processing, on the hand, will not distinguish between *far* or *closely disjoint* objects. So, while for many queries neighborhood-based preprocessing keeps *disjoint* relations, these *disjoint* relations are less common and, therefore, they tend to keep the evaluation cost low.

Table 6 shows the results of the searching process. We selected a random solution among the optimal solutions and a random solution among the solutions with worst confidence (i.e., larger number of constraints violated when they occur, or larger distance with respect to the query when no violation occur). A visual analysis of the results shows good matches, confirming that the forward-checking algorithm, which enforces that all constraints evaluated must be satisfied, is an appropriate choice for this experiment. The performance of this algorithm, however, decreases drastically depending on the number of occurrences of topological relations in the database.

6 Conclusions

This paper derived and compared composition-based and neighborhood-based pre-processing strategies for reducing the number of topological constraints that need to be satisfied in spatial query processing. The setting is tailored for similarity-based retrieval where a target configuration is either given by an existing spatial scene or derived from a sketch. Results of this study are that neighborhood-based ($V(g)$) pre-processing provides a good mechanism for reducing topological constraints that tends to reduce the computational cost of query evaluation. Although neighborhood-based pre-processing does not guarantee that topological constraints will be satisfied, solutions ranked by a similarity measure place optimal solutions first. Thus, similarity queries that are formulated in a visual language should exploit the metric characteristics of the configuration (i.e., distances between objects), even if only topological constraints are considered for making matches. In the case that topological queries are expressed with a command language [35], the composition-based pre-processing can only be used, as it always ensures that constraints that were not evaluated are satisfied.

In this work, pre-processing strategies were analyzed considering time and results of a similarity-based query process. The findings have implications on future work as they are useful not only for pre-processing queries, but for processing a whole database in order to create an indexing schema that organizes spatial interrelations between objects. Because only physically close interrelations are needed, one could drastically reduce the number of interrelations that need to be stored by a database. An aspect to be considered for future work is the potential of systematically selecting pre-processing strategies depending on the database and/or the query's characteristics. For example, constraints that were eliminated by both pre-processing strategies may indicate that there is a good chance of having a good balance between quality of

results and performance of query evaluation by using neighborhood-based preprocessing.

References

1. Egenhofer, M.: Query Processing in Spatial-Query-By-Sketch. *Journal of Visual Languages and Computing*. 8:(4) (1997) 403-424.
2. Papadias, D., Mantzouroguannis, M., Kalnis, P., Mamoulis, N., and Ahmad, I.: Content-Based Retrieval using Heuristic Search. *ACM-SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, (1999), 168-175.
3. Papadias, D., Mamoulis, N., and Delis, V.: Algorithms for Querying Spatial Structure. in Gupta, A., Shmueli, O., and Widom, J. (eds.), *24th VLDB Conference*, New York, NY, (1998), 546-557.
4. Rodríguez, A. and Godoy, F.: A Content-Based Approach to Searching Spatial Configurations. in Egenhofer, M. and Mark, D. (eds.), *GIScience 2002, Lecture Notes en Computer Science 2478*, Springer-Verlag, Berlin (2002), 260-275.
5. Rigaux, P., Scholl, M., and Voisard, A.: *Spatial Databases: with Application in GIS*. Academic Press, San Diego, CA (2002).
6. Egenhofer, M. and Frank, A.: LOBSTER: Combining AI and Database Techniques. *Photogrammetric Engineering & Remote Sensing*. 56:(6) (1990) 919-926.
7. Park, H.-H., Lee, C.-G., Lee, Y.-J., and Chung, C.-W.: Early Separation Filter and Refinement Steps in Spatial Query Optimization, *International Conference in Database Systems for Advanced Applications*, (1999), 161-169.
8. Clementini, E., Sharma, J., and Egenhofer, M.: Modeling Topological Relations: Strategies for Query Processing. *Computers and Graphics*. 18:(6) (1994).
9. Aref, A. and Samet, H.: Optimizing Strategies for Spatial Query Processing. *7th International Conference on Very Large Databases*, Barcelona, Spain, (1991), 81-90.
10. Samet, H. and Aref, W.: Spatial Data Models and Query Processing. *Modern Database Systems*, ACM Press (1995), 338-360.
11. Kriegel, H., Brinkhoff, T., and Schneider, R.: Efficient Spatial Query Processing. *IEEE Data Engineering Bulletin*. 16:(3) (1993) 10-15.
12. Egenhofer, M.: Deriving the Composition of Binary Topological Relations. *Journal of Visual Languages and Computing*. 5:(2) (1994) 133-149.
13. Blaser, A., *Sketching Spatial Queries*, Ph.D. Thesis. In Department of Spatial Information Science and Engineering, 2000, University of Maine: Orono, ME.
14. Florence, J. and Egenhofer, M.: Distribution of Topological Relations in Geographic Datasets,. *ACSM/ASPRS Conference*, Baltimore, MD (1996).
15. Papadias, D., Arkoumanis, D., and Karacapilidis, N.: On the Retrieval of Similar Configurations. in Poiker, T. and Chrisman, N. (eds.), *8th International Symposium on Spatial Data Handling*, International Geographical Union, Vancouver, Canada (1998), 510-521.
16. Papadias, D., Kalnis, P., and Mamoulis, N.: Hierarchical Constraint Satisfaction in Spatial Databases. *Proceeding of the Annual Meeting of the AAAI*, Orlando, FL, (1999), 142-147.

17. Meseguer, P.: Constraint Satisfaction Problems: an Overview. *AICOM*. 2:(1) (1989) 3-17.
18. Egenhofer, M. and Sharma, J.: Assessing the Consistency of Complete and Incomplete Topological Information. *Geographical Systems*. 1:(1993) 47-68.
19. Egenhofer, M. and Franzosa, R.: Point-Set Topological Spatial Relations. *International Journal of Geographical Information Systems*. 5:(2) (1991) 161-174.
20. Randell, D., Cui, Z., and Cohn, A.: A Spatial Logic Based on Regions and Connection. in Nebel, B., Rich, C., and Swarthout, W. (eds.), *Principles of Knowledge Representation and Reasoning, KR '92*, St. Charles, IL: Morgan Kaufmann, Cambridge, MA (1992), 165-176.
21. Egenhofer, M. and Al-Taha, K.: Reasoning About Gradual Changes of Topological Relations. in Frank, A., Campari, I., and Formentini, U. (eds.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Lecture Notes in Computer Science 639*, Springer-Verlag, Pisa, Italy (1992), 196-219.
22. Gaede, V. and Günther, O.: Multidimensional Access Method. *ACM Computing Surveys*. 30:(2) (1998) 170-231.
23. Egenhofer, M. and Shariff, A.: Metric Details for Natural-Language Spatial Relations. *ACM Transactions on Information Systems*. 16:(4) (1998) 295-321.
24. Godoy, F. and Rodríguez, A.: A Quantitative Description of Spatial Configurations. in Richardson, D. and van Oosterom, P. (eds.), *Spatial Data Handling*, Springer-Verlag, Ottawa, Canada (2002), 299-311.
25. Tarski, A.: On The Calculus of Relations. *Journal of Symbolic Logic*. 6:(3) (1941) 73-89.
26. Mackworth, A.: Consistency in Networks of Relations. *Artificial Intelligence*. 8:(1977) 99-118.
27. Maddux, R.: *Some Algebras and Algorithms for Reasoning about Time and Space*. Department of Mathematics, Iowa State University: Ames, IO, (1990)
28. Smith, T. and Park, K.: Algebraic Approach to Spatial Reasoning. *International Journal of Geographical Information Systems*. 6:(3) (1992) 177-192.
29. Tobler, W.: Cellular Geography. in Gale, S. and Olsson, G. (eds.), *Philosophy in Geography*, D. Reidel Publishing Company, Dordrecht, Holland (1979),
30. Toussaint, G.: The Relative Neighborhood Graph of a Finite Planar Set. *Pattern Recognition*. 12:(1980) 261-268.
31. Jaramcyzk, J. and Toussaint, G.: Relative Graph and their Relatives. *Proceedings of the IEEE*. 80:(9) (1992) 1502-1517.
32. O'Rourke, J.: *Computational Geometry*. Cambridge University Press, Cambridge, MA (1993).
33. Preparata, F. and Shamos, M.: *Computational Geometry: An Introduction*. Springer-Verlag, Berlin (1985).
34. Bacchus, F. and Grove, A.: On the Forward Checking Algorithm. in Montanari, U. and Rossi, F. (eds.), *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science 976*, Springer-Verlag, Berlin (1995), 292-309.
35. Egenhofer, M.: Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*. 6:(1) (1994) 86-95.