

A Content-Based Approach to Searching and Indexing Spatial Configurations

M. Andrea Rodríguez and Francisco A. Godoy

Department of Information Engineering and Computer Science
University of Concepción
and
Center for Web Research
University of Chile
Edmundo Larenas 215, Concepción, Chile.
{andrea,fgodoyf}@udec.cl

Abstract. A constant challenge of current spatial information systems is the retrieval of spatial configurations. This paper describes a new approach to retrieving spatial information whose novelty lies in using a content measure of topological relations to search and index spatial configurations. This approach uses a tree-based schema to index relations between objects' minimum bounding rectangles, it preprocesses the user query to obtain an ordered list of spatial constraints, and it explores three searching algorithms that work over an indexed domain: *full-restrictive* forward checking, *partial-restrictive* forward checking, and *permutation-based* searching. Experimental results show the viability of this type of indexing schema as well as the differences among searching algorithms.

1 Introduction

The importance of searching spatial information has been already recognized in a variety of disciplines, including image analysis, spatial and multimedia databases, geographic information systems, and digital libraries [1-3]. In such diverse applications possible queries can be as general as “find information related to the city of New York,” or may contain a more detailed specification of a user request, such as “find hospitals in New York City and adjacent cities” or “find images with two green circles separated by a blue one.” These queries can be expressed by visual languages, such as *VisualSeek* [4] and *Query by Sketch* [5, 6], or by verbal commands, e.g., the *extended SQL commands* [7].

In this work we are interested in answering queries that are composed of a set of spatial objects and a set of spatial relations for each pair of objects. These objects are referred to as variable objects of a query, whereas solutions to this query are the set of instances in the database that satisfies the query constraints. Studies addressing this type of query have taken a similarity-based approach to information retrieval by defining the set of spatial relations that can be used in a query, a similarity measure of

spatial relations, and a search algorithm for similarity retrieval [8-14]. Some of these studies are based on variations of 2D strings, which represent configurations as a sequential structure for each encoded dimension [12, 15, 16]. In a similar way, other studies have used a 3x3 matrix to determine the orientation relation by calculating the objects' proportional areas in the quadrants defined by the orthogonal projection of a reference object's MBR [17, 18]. Other studies represent configurations and queries by attribute relation graphs (ARGs) [2, 13, 19]. In these graphs, spatial relations are represented quantitatively by the distance and angle between centroids, and qualitatively by the symbolic representation of topological relations, such as the topological relations defined by Egenhofer and Franzosa [20, 21]. Similarity is then defined between quantitative values as the inverse of their difference [13, 19] and between qualitative relations as the inverse of the distance in a conceptual neighborhood structure [2, 22].

As opposed to image databases that consist of a large set of images, databases in Geographic Information Systems (GISs) are composed of a large number of spatial objects grouped into a few thematic layers, which usually cover the whole geographic space. In this context, the retrieval process consists of finding instances of objects in the database that satisfy the query constraints that involve spatial relations (i.e., topological, orientation, and distance relations) and objects' characteristics (e.g., semantic classification). In this way, the retrieval process is seen as a constraint satisfaction problem [23]. Considering query processing as a problem of constraint satisfaction, Papadias *et al.* [2, 10, 11] addressed retrieval of spatial configurations without restrictions on types and relations between objects.

This paper describes a general similarity-based approach to searching spatial configurations. Unlike previous works, this work explores a quantitative approach to index and search spatial relations. The work is based on a content measure that distinguishes topological relations by its asymmetric values for pairs of objects and that uses the relative size of objects as a metric refinement of the topological relations. Then, searching a solution uses a tree-based indexing schema over points within the space of possible values of the content measure. Following the ideas by Papadias *et al.* [2], we discuss searching algorithms that represent different degrees of satisfaction for query constraints: a *full-restrictive* or *hard* algorithm where each constraint must be satisfied, *partial-restrictive* or *soft* algorithm where some constraints may be violated, and a *permutation-based* or *semi-hard* algorithm where each constraint must be at least partially satisfied.

The organization of the paper is as follows. Section 2 presents the description of the content measure used by the similarity function and the indexing schema. Section 3 describes the preprocessing strategy. Section 4 introduces the three searching algorithms. Experiments using a randomly created data set are presented as a proof of concepts in Section 5. Conclusions and future research directions are in Section 6.

2. Framework for content-based searching and indexing

Searching spatial configurations is considered as a process of searching sets of instances in a database that satisfy query constraints expressed by spatial relations, in particular, topological relations. The main idea behind our approach to query processing is the determination of a content measure of topological relations and the use of this measure as basis for an indexing mechanism.

2.1 Content measure

This work uses the simplified and common representation of objects in current spatial indexing schemas for Geographic Information Systems (GISs); that is, objects' minimum bounding rectangles (MBRs). This simplification is widely used for its desirable computational properties, and it can be used as a first approximation in a similarity-based retrieval.

The defined content-measure of topological relations considers three basic primitives over objects' MBRs: (1) areas of individual MBRs and areas of intersection of pairs of MBRs, (2) diagonals of MBRs, and (3) minimum internal and external distances between boundaries of MBRs (i.e., $d_i(\square A, \square B)$ and $d_e(\square A, \square B)$, respectively) (Figure 1).

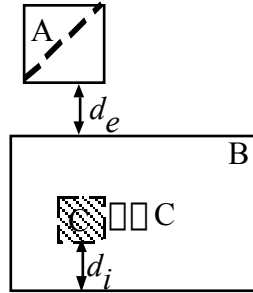


Fig. 1. Primitives of the content measure

For further clarification, we define a region's MBR and the intersection of pairs of MBRS as follows (Equations 1a-b):

$$MBR(A) = \{BOUND((x_1^A, y_1^A), (x_2^A, y_2^A)) / \{(x, y) \in A, \\ x_1^A \leq x \leq y_1^A \leq y \leq x_2^A \leq x \leq y_2^A \leq y\}\} \quad (1a)$$

$$INTSERSECT(MBR(A), MBR(B)) = \{BOUND((x_1, y_1), (x_2, y_2)) / \\ \{(x, y) \mid x_1 \leq x \leq y_1 \leq y \leq x_2 \leq x \leq y_2 \leq y, (x, y) \in A \cap (x, y) \in B\}\} \quad (1b)$$

The underlying idea of this measure is that *distance* between objects is a basic parameter for the refinement of *disjointness*, while the *area* of the objects is a basic parameter for the refinement of *overlapping* (Equation 2) [24].

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

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

$$\text{distance}(\square A, \square B) = \begin{cases} d_e(\square A, \square B) & \text{if } A \cap B = \emptyset \\ d_i(\square A, \square B) & \text{if } A \cap B \neq \emptyset \end{cases}$$

This measure is asymmetric; so, describing the arrangement of two objects needs two values, one in each direction of the relation. Figure 2 presents the range of values in 2D that characterizes the topological relations between MBRs. The boundaries of the regions in this figure were determined by considering extreme cases and then creating the corresponding parametric equations. As Figure 2 reflects, all eight topological relations between two extended objects, such as those defined by the 9-Intersection model [20] and the RCC model [25], can be defined in the 2D space that maps values of the content measure.

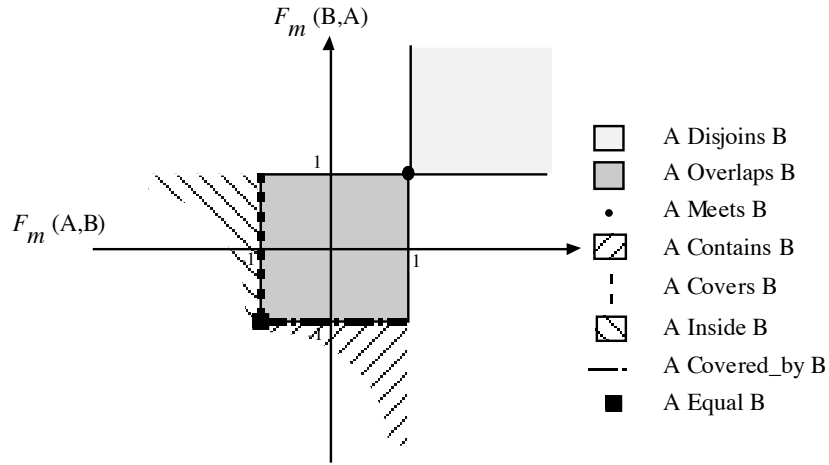


Fig. 2. Domain values for the content measure classified into 8 topological relations

2.2 Indexing schema

The idea of an indexing schema is to avoid the exhaustive review of the whole dataset in a searching process. Geographic databases deal with large numbers of objects, and user queries in these systems contain a variable number of objects. A traditional

approach to this problem has been to use a spatial indexing schema over MRBs with heuristics to guide the search process of objects with specific topological constraints [2, 10, 11]. The approach of this work, instead, uses an index of spatial *relations between objects* rather than an index of the spatial *locations of objects*. Although indexing spatial relations may mean generating an index structure that is larger than the index structures based on spatial location, retrieval processes that demand the constraint satisfaction of spatial relations can be strongly improved as we avoid the constant evaluations of topological relations that occurs with a traditional indexing schema.

With the content measure defined above, an indexing schema can be easily derived from any spatial access methods. For example, a tree-based indexing schema helps to organize the space into divisions of spatially close points, where points in this space represent the asymmetric values of the content measure between pairs of objects' MBRs. In Figure 3 an example of a R-tree [26] is shown.

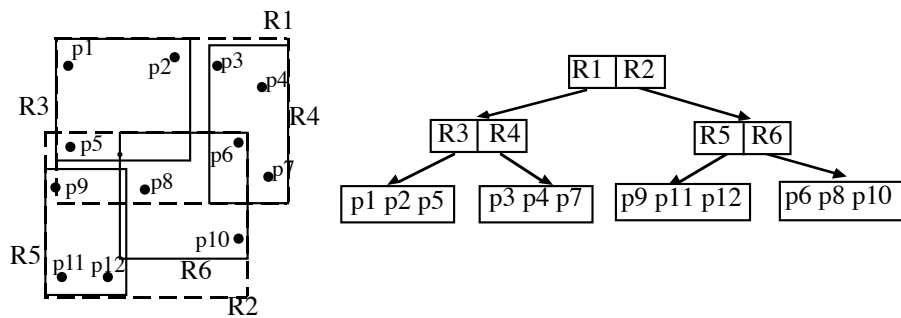


Fig. 3. Tree-Based indexing schema

One might consider using a different indexing schema that organizes configurations as a whole, using, say, a multidimensional vector space, but the fact that it is impossible to know in advance the number of variable objects in a user query makes it impossible to predefine a vector space with fixed number of dimensions. Furthermore, our indexing schema exploits the metric refinement of topological relations in such a way that it is impossible to handle just eight topological relations between regions as an array of 8 values. The defined content measure distinguishes an infinite number of variations of these topological relations given by the metric refinement of areas, diagonals, and distances between MBRs.

The total number of possible binary spatial relations of a fixed kind (e.g., *inside*) among n objects is $n(n-1)$. In order to reduce the number of relations to be indexed, we consider that *non-disjoint* relations are relevant, because they indicate physical connection between objects [6, 27]. So, we eliminate *disjoint* relations with a relative distance separation larger than d (i.e. $F_m () > d$), a parameter that is set before the indexing process starts.

The search for a relation (i.e., a query constraint) in the trees uses a distance function between a query constraint and the spatial relation of instances in the database. In this evaluation, a query constraint is represented by the content measure of two variable objects of the query v_i and v_j , that is $R(v_i, v_j)$, and a possible solution is represented by the content measure of two instances u_k , and u_l in the database, that is $R(u_k, u_l)$, taking each direction of the relation separately (Equation 3). Both variable objects and instances are presented by their MBRs.

$$\begin{aligned} d(R(v_i, v_j), R(u_k, u_l)) &< \sqcap R(v_i, v_j) \\ d(R(v_j, v_i), R(u_l, u_k)) &< \sqcap R(v_j, v_i) \end{aligned} \quad (3)$$

A constraint is said to be satisfied when the distance is less than a threshold function $\sqcap R(v_i, v_j)$, which depends on the relation (i.e., constraint) that the system is searching for. The idea of this threshold function is that distance is not homogenous in the space of the content measure. For example, the difference between two *disjoint* relations is not as relevant as the difference between a *meet* and *overlap* relations. Break values in the space of the content measure are 1 and -1 (Figure 2). The value 1 separates *disjoint* from *non-disjoint* relations, whereas the value -1 separates *inside* from *overlap*. So, we have defined a threshold function that considers the distance between these break values and the query constraint (Equation 4). Thus, as the absolute value of the content measure that describes a query constraint approaches the value 1, the difference becomes more relevant than when the constraint represents a *disjoint* or *inside* relation with content values distant from 1 and -1, respectively.

$$\sqcap R(v_i, v_j) = \text{abs}(1 \sqcap \text{abs}(R(v_i, v_j))) \cdot a, \quad a < 1.0 \quad (4)$$

3 Query Pre-processing

Query preprocessing plays two important roles: (1) eliminating irrelevant or implicit relations and (2) sorting the list of constraints by relevance or short searching path. As in the indexing process, we eliminate *disjoint* relations with a distance separation larger than a threshold (d), and we check for implicit relations through composition tables of topological relations [21]. A path consistency algorithm [28] can check whether or not by eliminating relations we can keep the query consistent. For example, Figure 4 presents a configuration with 3 objects, so 6 spatial relations can be determined: *A meets B*, *A contains C*, *B disjoint C*, and their respective converse relations. By definition of the converse relations we could eliminate 3 explicit relations that are completely derived from their converse. Using the composition operation, we can then derive the relation *B disjoint C* by applying the following composition:

$$B \text{ meets } A ; A \text{ contains } C \sqcap B \text{ disjoint } C$$

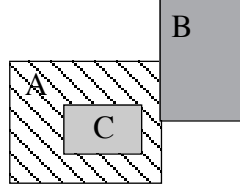


Fig. 4. Configuration with derivable relations

Applying the criterion that *non-disjoint* relations are more relevant [6, 27], we create an increasing ordered list of spatial constraints based on the values of the content measure. Small values of the content measure correspond to large degrees of *overlapping*, ranging between *equal*, *inside*, *covered_by*, *overlap* and *meet* relations.

4 Searching Algorithms

Three different algorithms were developed to search configurations using the indexing schema defined above. All three algorithms consider a forward checking strategy. While the two first algorithms force individual constraints be satisfied, the last algorithm relaxes this condition and checks a global similarity value in each forward checking step.

The general structure underlying these algorithms is defined by sets of instances in the databases that satisfy the constraints expressed in the query. Thus, given a user query with a set of n variable objects $Q = \{v_1, \dots, v_n\}$, each configuration in the set of possible solutions has a structure with n instances $S = \{u_1, \dots, u_n\}$ in the database. The goal of the tree algorithms defined below is not to find the first solution, but rather to create a set of solutions using a ranking schema. The ranking of solutions is determined by the sum of distances between content measures calculated for each pair of variables in the query and the corresponding pair of instances in the database (Equation 5). This function represents a global similarity of configurations with respect to a user query.

$$D(Q, S) = \prod_{v_i, v_j \in Q; u_i, u_j \in S} \sqrt{\left(F_m(v_i, v_j) \square F_m(u_i, u_j)\right)^2 + \left(F_m(v_j, v_i) \square F_m(u_j, u_i)\right)^2} \quad (5)$$

In searching instances for each constraint, the algorithms access the index tree whose depth determines the number of checking operations; that is, the complexity of this algorithm.

4.1 Full restrictive.

The full restrictive algorithm checks constraints one-by-one using the index schema. As it finds solutions for each constraint, it performs a *join* operation with the previously found solutions to match instances (Figure 5). A join operation performs a Cartesian product of its two arguments (i.e., previous solutions and solutions for the last constraint analyzed), performs a selection forcing equality on those instances that should correspond to equivalent variable objects in the query, and removes duplicates.

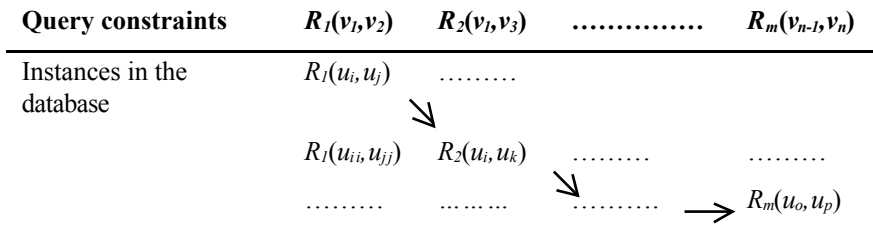


Fig. 5. Full restrictive strategy to searching configurations

In addition to the complexity given by the search in the index structure, this algorithm needs to perform m join operations, where m is the number of constraints. For each join operation, the algorithm checks l times whether the instances match or not the previous candidate solutions, where l is the smallest number between the number of solutions of the last constraint analyzed and the number of previously found solutions. The final ranking of solutions is given by the global similarity measure defined in Equation 5.

4.2 Partial restrictive

The partial restrictive algorithm finds configurations where some of the constraints may not be satisfied. Like the full-restrictive algorithm, this algorithm finds configurations with instances that satisfy the constraints, even when the solutions may contain a smaller number of objects than the original user query. This may happen when few constraints are satisfied and none of these constraints involves instances for a variable object of the query. The fact that some constraints may not be satisfied, however, does not imply that the solutions will always contain a smaller number of instances than the number of objects in the user query. For example, a query with three objects involves three pairs of content measures (one for each pair of objects). If two constraints are satisfied it will result in solutions with 3 objects.

The types of operations for this algorithm are equivalent to the operations of the previous algorithm. Unlike the previous algorithm, however, when instances do not match, the algorithm does not necessarily eliminate the candidate solution. The best solutions in this algorithm are the ones with a larger value of global similarity

(Equation 5) and with a larger number of instances in the database assigned to variable objects in the query.

4.3 Permutation-Based Search

The permutation-based search finds configurations by looking for global similarity rather than for individual constraint satisfactions. In this way, constraints may not be completely satisfied; however, the search finds solutions that are expected to be the most similar configurations to the user request. In the worst case, finding the most similar configuration implies all n -permutations of N objects, with n being the number of objects in the query and N the number of objects in the database. Thus, for $N \gg n$, the retrieval process is exponential in the size of the query ($O(N^n)$). To avoid the exhaustive search, this algorithm reduces the domain of searching to the sets of objects that satisfy each constraint, obtained using the indexing schema. Once these objects are found, the algorithm makes the needed permutations. The types of operations for this algorithm are the search in the index schema and permutations, considering a global similarity value that is larger than a given threshold.

The implementation of this algorithm follows. A query (Q) is represented by a list of variable objects (variables v_i). Based on this list of objects, a list of constraints (C) is obtained. For each constraint, the algorithm searches for solutions (S) using the tree-based schema. Each solution is also represented by a list of pairs of instances in the database. Then it takes each previously found global solution ($GS[]$), that is, a set of $n \geq 2$ instances in the database, and permutes its instances with the instances found in the last search, resulting in a new candidate solution SS . If this new solution gives a global similarity (Equation 5) with respect to the user query that is larger than a given threshold, the solution is kept and added to the set of global solutions. This process is repeated until all constraints have been sequentially considered.

Procedure Permutation_Based_Similarity (Q)

```

C [] Get_Constraints(Q); GS [] nil
for i [] 1 to m do //for each constraint
    temp [] nil; X[] Content_Measure(C[i]); S[] Search_Tree(X)
    if i = 0 then Add_Solutions(S,GS) else
        for j [] 1 to Size(GS) do //for each global solution found previously
            for k [] 1 to Size(S) do //for each solution to the current constraint
                SS[] Permute(GS[j],S[k])
                if (D(Q,SS) [] i*[]) then Add_Solutions(SS,temp)
            GS[] temp
        Sort(GS); return GS
end Permutation_Based_Similarity

```

5 Experimental Results

In order to test our indexing schema and algorithms, we have created a database with 18,000 random objects of 6 different semantic classes. In this database, *disjoint* is the relation with the highest frequency of occurrences (90%), followed by the *overlap* (7%) and *contain* relations (3%). The frequency distribution of relations is due to the fact that the database is composed of objects' MBRs that are homogenously distributed over one large space. Thus, there is not a concentration of objects in particular areas of the space, which could produce a larger number of *overlap* relations. In addition, two real objects that *meet* most likely results in their MBRs *overlapping* rather than *meeting* so, the number of *meet* relations is small.

A R-tree structure [26] was created as an index of content measures for combinations of object classes, giving rise to 21 different trees. These 21 different trees result from the symmetric combination of 6 classes plus the combination of the 6 classes with themselves, that is, $6*5/2 + 6 = 21$. Nodes in each of these trees contain between 250 and 500 elements, given a maximum depth of 3 for each tree. The total number of relations indexed was 150,000, as the result of the elimination of relations whose content measures in both directions were larger than 4.0 (i.e., $d = 4.0$). The size of the spatial indexes was 50% larger than the size of the stored objects' MBRs.

Three queries are presented here: a configuration existing in the database (Q1) and two non-existing configurations (Q2 and Q3). All three examples contain four objects and three different object classes. The existing configuration (Figure 6) contains an *overlap* relation between objects of different classes. The rest of the relations are all *disjoint*.

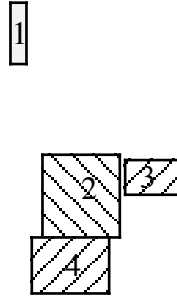


Fig. 6. Example of existing configuration (Q1)

Two queries that were not exactly equivalent to any configuration in the database were defined. The first of these queries (Q2) involves *overlap* relations, whereas the second one (Q3) involves just *disjoint* relations. Since *disjoint* is the most common relation in the database, this type of query may imply a larger number of comparisons

between candidate instances than comparisons for those queries with rare relations in the database.

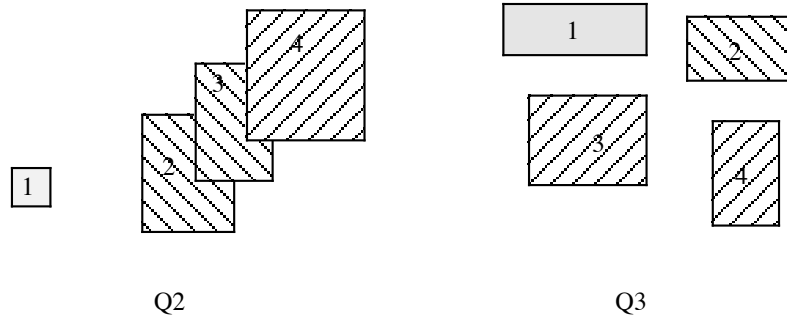


Fig. 7. Examples of non-existing configurations (Q2, Q3)

After defining the queries, the system performs a preprocessing to eliminate and sort constraints. As a result of this preprocessing, no constraint was eliminated from the queries, since the *disjoint* relations have a short separation between MBRs ($d \leq 4.0$), and composition could not derive any relation in these queries. Based on the sort operation over content measures, the order of constraints taking in the search process is as follows:

Table 1. Ordered constraints by query (where constraints are expressed by the pair of objects that defines the relation)

Query	List of Ordered Constraints
Q1	{(2,4), (2,3), (4,3), (1,2),(1,3),(1,4)}
Q2	{(3,2),(3,4),(2,4),(1,2),(1,3),(1,4)}
Q3	{(1,3),(1,2),(2,3),(2,4),(3,4),(1,4)}

As Table 1 indicates, *overlap* relations take precedence over *disjoint* relations. In particular, by looking at query Q2 it is possible to notice that the system first searches for the three *overlapping* objects, which indeed are less frequent than the *disjoint* objects in the database.

For the searching process, parameter a of Equation 4 was set to 0.5 and the *threshold* for the permutation-based algorithm was set such that the sum of distances between relations of the query and solutions is less or equal to the number of constraints (Equation 5). Results of query Q1 are shown in Figure 8. These results are consistent across algorithms, since these algorithms find as best solution the configuration equivalent to the query. The last algorithm (i.e., permutation-based) finds three solutions based on the threshold used for filtering results. The two less

similar solutions consist of one and two different objects, respectively, with respect to the equivalent result.

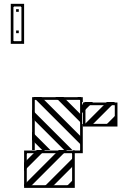
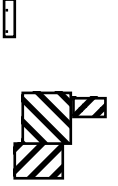
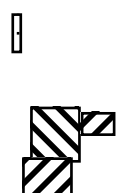
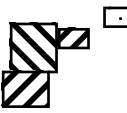
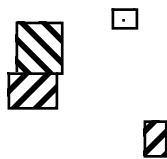
Q1	Ranking: 1°	Ranking: 2°	Ranking: 3°
Alg. 1			
Alg. 2			
Alg. 3			

Fig. 8. Results of first query Q1

Results of query Q2 are shown in Figure 9. For this query only algorithm 2 (i.e., partial restrictive) finds a solution. The unique found solution has a missing object, which is the object 3 that overlaps objects 2 and 4. An intuitive analysis suggests that although it is useful to have partial solutions, these solutions should have contained the objects and constraints that constitute the main features of the user request. In particular for query Q2, the constraints of objects (2,3) and (3,4) seem to have more relevance than the constraints that involve object 1. This is considered in the order of evaluations; however, the relevant constraints are not forced to be satisfied in algorithm 2.




Q2	Ranking: 1°	Ranking: 2°	Ranking: 3°
Alg. 2		 	

Fig. 9. Results of first query Q2

Results of query Q3 are shown in Figure 10. Both algorithms 2 and 3 (i.e., partial restrictive and permutation-based, respectively) find solutions. Since algorithm 2 finds solutions while algorithm 1 does not, it is possible to conclude that not all 6-constraints are satisfied. Algorithm 1 should always give a subset of solutions of algorithm 2. Algorithm 3 finds a solution that, as a global similarity evaluation, gives a good result. The result given by the last algorithm satisfies all topological relations among objects of the user query, but it is swapped with respect to the vertical axis. This is expected since our content measure is independent of scale, global rotation or swapping. The fact that algorithm 3 finds a good solution while algorithms 1 and 2 do not is due to the restrictive error tolerance given in Equation 3.


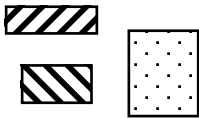
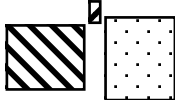
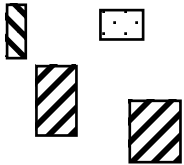
Q3	Ranking: 1°	Ranking: 2°	Ranking: 3°
Alg. 2			
Alg. 3			

Fig. 10. Results of first query Q3

As we explained before, an exhaustive search of query constraints would imply all 4 permutations (4 variable objects) of 18,000 objects in the database, which is of order $O(18,000^4)$. In order to compare the complexity of algorithms, we have made an experimental calculation of the number of comparisons for each of the queries and each of the algorithms. We do not give CPU time, because we consider it to be dependent on computational resources. Instead, we use these comparisons that do not include the search in the index tree, since this search is equivalent in all three algorithms. Table 2 shows the number of comparison for each query using each algorithm.

Table 2. Number of comparisons for each query using each algorithm

Query	Algorithm	Number of Comparison
Q1	Full restrictive	7431
	Partial restrictive	7431
	Permutation-based	184341
Q2	Full restrictive	560
	Partial restrictive	12684
	Permutation-based	1120
Q3	Full restrictive	55661
	Partial restrictive	57810
	Permutation-based	207998

The number of comparisons has a direct relation to the query and database characteristics. Queries Q1 and Q3 have more solutions than query Q2. Since Q1 and Q3 include most *disjoint* relations, there are many candidate solutions for individual constraints and, therefore, the number of *join* as well as *permutation* operations is large. Query Q2, on the other hand, represents a rare combination of constraints for instances in the database, such that the number of solutions is quickly reduced after a *join* or *permutation* operation. Indeed, just the *partial restrictive* algorithm finds solutions for this query.

In the previous experiments we do not provide comparisons with other algorithms, since to the best of our knowledge, none of the previous studies have attempted to define a continuous content measure that distinguishes topological relations. In addition, previous studies have used traditional indexing schemas of objects' MBRs instead of indexing spatial relations. In detail, using our R-tree index of spatial relations, in the best case the search process visits a number of nodes equivalent to the depth of the tree, that is, 3 for our experiments. Indeed, for our experiments, none of the searches in the index took more than 27 visits. If we consider that each of our nodes does not have more than 500 children, we have compared less than $27 \cdot 500 =$

13,500 value pairs to obtain solutions for each constraint. If we perform an exhaustive revision of the whole space to find a pair of objects that satisfies a query constraint, the systems should have checked $O(18,000^2)$ combinations.

6 Conclusions

This work has presented a new approach to searching configurations in spatial databases. It defines a content measure that is independent of scale and that characterizes the topological distribution of spatial objects. This content measure constitutes a metric refinement of topological relations that depends on objects' relative sizes and relative locations. This work uses this content measure as basis for an indexing schema and a similarity function to search spatial configurations. The search process is implemented with three different algorithms (i.e., full restrictive, partial restrictive, and permutation-based algorithms), which provide different degrees of satisfaction, that is, solutions where all constraints are satisfied, solutions where at least some of the constraints are satisfied, and solutions where an overall similarity is fulfilled.

The indexing schema over spatial relations can strongly improve any search that focuses on structural aspects of spatial configurations. For queries that have equivalent configurations in databases, all three algorithms give the same results, with the permutation-based algorithm being the one that is computationally more demanding than the other two. In cases where there are no configurations equivalent to the query, the partial restrictive and permutation-based algorithms have more chances to obtain solutions. The partial restrictive algorithm has better chances of giving good solutions if we consider that some of the constraints must be satisfied to keep the prominent characteristics of the user request. In other cases, the permutation-based algorithm provides a good alternative, though with additional computational cost.

As future work, we would like to extend the content measure to objects in 3D. In the context of algorithms, we want to explore the concepts of genetic algorithms [29] to give flexibility in the search of most similar configurations by avoiding local solutions that may arise when using a forward checking approach. The satisfaction of individual constraints does not guarantee that the solutions are the best solutions. For query preprocessing, we would like to include the concept of *mandatory* query constraints, such that some of the constraints must or may be satisfied. For example, given the results of query Q2, we could have imposed that, in order to accept the answer, the *overlapping* constraint must have been satisfied. An interesting aspect for future consideration is how this index is affected by changes in the locations of objects. Although changes in continuous locations produce continuous changes in the values of the content measure, we have not analyzed with enough depth the effect of these changes on the indexing schema.

7 Acknowledgments

This work has been funded by CONICYT - Chile, under Fondecyt grant 1010897, and by Millenium Nucleous Center for Web Research, grant P01-029-F, Mideplan, Chile.

References

1. Berchtold, S., Bohm, C., Braunmuller, B., Keim, D., and Kriegel, H.-P.: Fast Parallel Similarity Search in Multimedia Databases. ACM SIGMOD International Conference on Management and Data, (1997)
2. 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, Vancouver, (1998) 510-521
3. Wang, J., Li, J., Chan, D., and Wiederhold, G.: Semantics-Sensitive Retrieval for Digital Picture Libraries. Digital-Library Magazine **5**:(11) (1999)
4. Smith, J. and Chang, S.-F.: Integrated Spatial Query and Feature Image Query. Multimedia Systems **7** (1999) 129-140
5. Egenhofer, M.: Query Processing in Spatial-Query-By-Sketch. Journal of Visual Languages and Computing **8**:(4) (1997) 403-424
6. Blaser, A.: Sketching Spatial Queries. In Department of Spatial Information Science and Engineering, University of Maine, U.S.A, (2000)
7. Egenhofer, M.: Spatial SQL: A Query and Presentation Language. IEEE Transactions on Knowledge and Data Engineering **6**:(1) (1994) 86-95
8. Papadias, D., Mamoulis, N., and Theodoridis, Y.: Constraint-Based Processing of Multiway Spatial Joins. Algorithmica. Special Issue on Algorithms for GIS (2001)
9. Papadias, D.: Hill Climbing Algorithms for Content-Based Retrieval of Similar Configurations. ACM Conference on Information Retrieval, Athens (2000)
10. 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 (1999)
11. Papadias, D., Mamoulis, N., and Delis, V.: Algorithms for Querying Spatial Structures. 24th VLDB Conference, New York, NY (1998)
12. Lee, S. and Hsu, F.: Spatial Reasoning and Similarity Retrieval of Images Using 2D C-Strings Knowledge Representation. Pattern Recognition **25**:(3) (1992) 305-318
13. Berretti, S., Bimbo, A.D., and Vicario, E.: The Computational Aspect of Retrieval by Spatial Arrangement. International Conference on Pattern Recognition, (2000)
14. El-Kwae, E. and Kabuka, M.: A Robust Framework for Content-Based retrieval by Spatial Similarity in Image Databases. ACM Transactions on Information Systems **17**:(2) (1999) 174-198
15. Castaglia, G., Tortora, G., and Arndt, T.: A Unifying Approach to Iconic Indexing for 2-D and 3-D Scenes. IEEE Transactions on Knowledge and Data Engineering **4**:(3) (1992) 205-222
16. Lee, S., Yang, M., and Chen, J.: Signature File as a Spatial Filter for Iconic Image. Journal of Visual Language and Computing **3** (1992) 373-392

17. Goyal, R. and Egenhofer, M.: Similarity of Direction Relations. In Jensen, C., Schneider, M., Seeger, B., and Tsotras, V. (eds.): Seventh International Symposium on Spatial and Temporal Databases. Lecture Notes in Computer Science, Vol. 2121, Springer-Verlag, Berlin Heidelberg New York (2001) 36-55
18. Goyal, R. and Egenhofer, M.: Cardinal Directions Between Extended Spatial Objects. IEEE Transactions on Knowledge and Data Engineering (in press)
19. Petrakis, G. and Faloustos, C.: Similarity Searching in Medical Image Databases. IEEE Transactions on Knowledge and Data Engineering **9**(3) (1997)
20. Egenhofer, M. and Franzosa, R.: Point-set topological spatial relations. International Journal of Geographical Information Systems **5**(2) (1991) 161-174
21. Egenhofer, M.: Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing **5**(2) (1994) 133-149
22. Bruns, T. and Egenhofer, M.: Similarity of Spatial Scenes. In Kraak, M. and Molenaar, M. (eds.): Seventh International Symposium on Spatial Data Handling (SDH '96), Delft, The Netherlands, (1996) 4A.31-42
23. Meseguer, P.: Constraint Satisfaction Problems: an Overview. AICOM **2**(1) (1989) 3-17
24. Godoy, F. and Rodríguez, A.: A Quantitative Description of Spatial Configurations. Spatial Data Handling, Ottawa, Canada (2002)
25. 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, Morgan Kaufmann, Cambridge, MA, (1992) 165-176
26. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. ACM SIGMOD Int. Conf. on Management of Data, (1984)
27. Florence, J., Prediction Frequency of Topological Relations in Geographic Datasets. In Department Spatial Information Science and Engineering, University of Maine (1997)
28. Egenhofer, M. and Sharma, J.: Assessing the Consistency of Complete and Incomplete Topological Information. Geographical Systems **1** (1993) 47-68
29. Chambers, L.: The Practical Handbook of Genetic Algorithms: Applications. Chapman & Hall (2000)