Processing Fuzzy Spatial Queries : A Configuration Similarity Approach

DIMITRIS PAPADIAS
Department of Computer Science, Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong
email: dimitris@cs.ust.hk

NIKOS KARACAPILIDIS
Department of Computer Science, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
email: karacapi@di.epfl.ch

and DINOS ARKOUMANIS
Department of Electrical and Computer Engineering, National Technical University of Athens, Greece
email: dinosar@dbnet.ntua.gr

Abstract. Increasing interest for configuration similarity is currently developing in the context of Digital Libraries, Spatial Databases and Geographical Information Systems. The corresponding queries retrieve all database configurations that match an input description (e.g., "find all configurations where an object x_0 is *about 5km northeast* of another x_1, which, in turn, is *inside* object x_2"). This paper introduces a framework for configuration similarity that takes into account all major types of spatial constraints (topological, direction, distance). We define appropriate fuzzy similarity measures for each type of constraint to provide flexibility and allow the system to capture real-life needs. Then we apply pre-processing techniques to explicate constraints in the query, and present algorithms that effectively solve the problem. Extensive experimental results demonstrate the applicability of our approach to images and queries of considerable size.

1. INTRODUCTION

As opposed to visual object similarity, which is based on visual features of objects (e.g., shape, size, texture, colour), *configuration similarity* refers to their arrangements in space. Related queries retrieve all images in the database that match some input configuration which is expressed by a set of binary *direction* (e.g., north), *topological* (e.g., overlap, inside) or *distance* constraints (e.g. about 1 km away). The significance of configuration similarity for content-based retrieval has already been recognised in a variety of disciplines including Image, Spatial and Multimedia Databases and Geographic Information Systems. As a consequence, a number of verbal and pictorial languages has been proposed to accommodate such queries (Chang et al., 1987; Papadias and Sellis, 1995; Egenhofer, 1997; Agouris et al., 1998). Increasing interest for configuration similarity is also developing in the context of Digital

Correspondence should be addressed to Dimitris Papadias: http://www.cs.ust.hk/faculty/dimitris/, tel: +852-2358-6971, fax: +852-2358-1477

Libraries as a complement to traditional Information Retrieval techniques for text (Smith and Chang, 1996).

Most database approaches on similarity retrieval have focused on narrow domains and applied specific algorithms to match particular application needs. For instance, Bach et al. (1993) apply exhaustive search for the retrieval of facial images, while Rabbiti and Savino (1992) propose a multilevel *signature* technique for images where the number and the type of objects in all images are known in advance. Some approaches (e.g., Orphanoudakis et al., 1994; Chu et al., 1994; Petrakis and Faloutsos, 1997) deal with similarity of medical images to support clinical decision making.

Other researchers have developed more general purpose techniques which may not take full advantage of the special structure of a given domain, but are applicable to a wider variety of applications. For instance, Lee and Hsu (1992) use symbolic object *projections* encoded in 2D strings and string matching algorithms to retrieve configuration similarity; Nabil et al. (1996) propose another projection-based technique that uses conceptual neighbourhoods to measure relation similarity; Sistla et al. (1995) present a set of direction and topological relations in 3D space and similarity measures based on this set. Unlike previous approaches, where spatial relations are defined on extended objects, Gudivada and Raghavan (1995) use the angles between object centroids to define image similarity based on directions. An advantage of this approach is that it can model rotational transformations of images in arbitrary angles, while projection-based methods can only handle rotations of $n\pi/2$ (rotation is also studied in Tagare et al., 1992).

The majority of the above methods only deal with cases where images are re-arrangements of the same set of objects. The queries simply retrieve all images where some configuration of *specific objects* is satisfied (e.g., "find all images where object A is *above* object B"). This paper focuses on the general problem of configuration similarity where images contain *arbitrary objects* and the queries refer to *object variables* rather than instances. This is, in general, a hard exponential problem as indicated by early studies on Computer Vision (Ballard and Brown, 1984). In order to provide a solution as general as possible, we consider *extended objects* and *centroids* since both approaches have some advantages. More specifically, centroids are more appropriate for rotations and distances, while extended objects are needed for topological relations.

In addition, our approach can deal with the inherent uncertainty of spatial information processing. There may be three types of uncertainty in real-life applications: (i) the first refers to "*fuzzy objects*", that is, objects that do not have well defined boundaries (e.g., residential areas, forests), or have boundaries that change over time (e.g., shorelines in the presence of tide)[1]; (ii) even if objects are rigid, they may be related by "*fuzzy relations*", i.e., a pair of objects may belong to multiple relations with different grades of membership (e.g., Greece can be viewed as being both *east* and *south-east* of Italy); (iii) the third type

[1] There has been a significant amount of work on modelling objects with fuzzy boundaries both in Geography and Surveying Engineering (Burrough and Frank, 1996) and in Computer Science (Topaloglou, 1996).

of fuzziness arises from ambiguous interpretations of spatial relations, that is, different users may imply different relations when using the same linguistic term[2].

In this paper we deal mainly with the last two types of uncertainty. Fuzzy sets theory (Zadeh, 1965) provides a useful framework to deal with problems characterized by imprecision due to subjective and qualitative evaluations, as in configuration similarity. The concept of approximation using fuzzy values has been well addressed in fuzzy set theory (Zimmermann, 1991), while a variety of measures of similarity for fuzzy sets has been already proposed (e.g., Pappis and Karacapilidis, 1993). Our contribution consists of: (i) the proposal of a framework for configuration similarity which includes all major types of spatial constraints and handles efficiently the fuzziness of the problem; (ii) the development of effective query pre-processing methods, and (iii) the implementation of appropriate retrieval algorithms.

The paper is organized as follows: Section 2 describes the *set of spatial relations* allowed in the expression of queries and defines *measures of similarity* between images based on these relations. Section 3 formally defines the problem of configuration similarity and introduces three *types of retrieval*. Section 4 presents *pre-processing techniques* that significantly improve performance, while Section 5 illustrates the proposed *algorithms* for similarity retrieval. Finally, Section 6 describes the *experimental results* and Section 7 concludes the paper with future work directions.

2. SPATIAL SIMILARITY DEFINITIONS

Most researchers in Spatial Databases and GISs, have dealt with three main classes of spatial constraints: topological, direction and distance. We consider all types of constraints here and use fuzzy similarity (Dubois at al., 1993) to capture the uncertainty which is inherent in most spatial applications.

2.1. Topological Constraints

Topological constraints express the concepts of inclusion and neighbourhood. A large body of the related work has focused on the *intersection* model (Egenhofer and Franzosa, 1991) which describes relations using intersections of object's interiors and boundaries. The model defines the following set of 8 pairwise disjoint topological relations between planar regions:

$$T=\{\textbf{D}isjoint, \textbf{M}eet, \textbf{O}verlap, Co\textbf{V}ers, \textbf{C}ontains, \textbf{E}qual, Covered\_\textbf{B}y, \textbf{I}nside\}.$$

Figure 1 illustrates these relations in the form of a conceptual neighbourhood graph (Egehnofer and Al-Taha, 1992; Hernandez, 1994). Nodes in the graph denote relations that are linked through an edge if they can be directly transformed to each other by continuous deformations (enlargement, reducement, movement). For instance, starting from relation *disjoint* and extending (or moving) one of the objects, we

[2] A study about the - sometimes ambiguous - use of spatial relations in natural language can be found in (Mark and Egenhofer, 1994).

derive relation *meet*. With a similar extension we can get the transition from *meet* to *overlap* and so on. *Disjoint* and *overlap* are called 1st degree neighbours of *meet*. Depending on the allowed deformation and the relations of interest, several graphs may be obtained (e.g., Bruns and Egenhofer, 1996; Nabil et al., 1996).

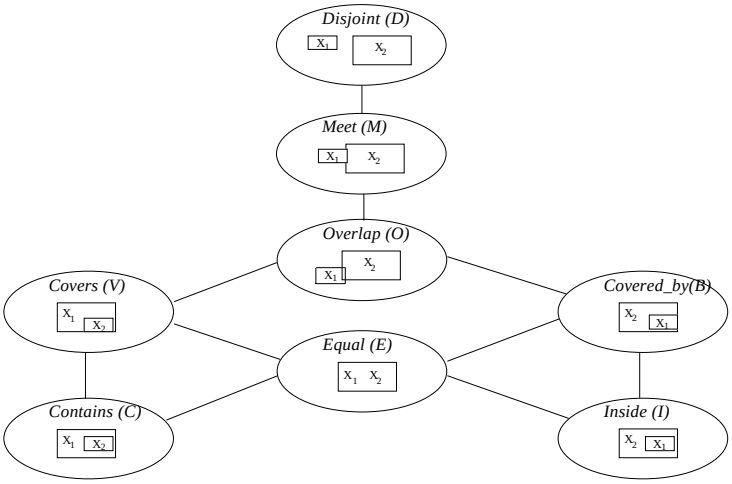


Figure 1 Topological relations

We use the distance between two relations in the graph to define their similarity measure. If T_i and T_j are two topological relations (T_i, $T_j \in T$), their *topological similarity measure* σ_T is defined as follows[3]:

$$\sigma_T(T_i,T_j)=\begin{cases} 1 & \text{if } T_i = T_j \\ \tau\ (0<\tau<1) & \text{if } T_i \text{ and } T_j \text{ are 1st degree neighbors} \\ 0 & \text{otherwise} \end{cases}$$

A topological constraint C_T is a disjunction (set) of topological relations of T. The similarity between C_T and a relation T_j is the maximum similarity between T_j and any relation in C_T:

$$\sigma_T(C_T,T_j)=\max_{\forall T_i \in C_T} \sigma_T(T_i,T_j).$$

If, for instance, C_T is *disjoint*$\vee$*meet* ({D,M}) and T_j =*overlap*, then $\sigma_T(C_T,T_j)$= max(σ_T(D,O),σ_T(M,O))=τ. U_T denotes the *universal topological constraint* and it corresponds to the disjunction of all relations: U_T= {D, M, O, V, C, E, I, B}. It is used to represent that the topological relation between two objects is unconstrained. The similarity of U_T to any topological relation is 1.

2.2. Direction Constraints

It has been suggested (e.g., Frank, 1996) that people manipulate concrete relations rather than continuous angles to express and reason about directions. Most previous work defines directions using either object projections (e.g., Sharma, 1996; Papadias and Egenhofer, 1997) or centroids (e.g., Hernandez, 1994). Each approach has its advantages and shortcomings; for a detailed discussion see (Frank, 1996). We

[3] This is one of the many possible definitions (it is used in the experiments of Section 6). Alternative ones may be preferable for some application domains.

follow a centroid-based method, where the direction between two objects is determined by the angle between their centroids. We use the following set of cardinal direction relations:

$$A=\{\textbf{\textit{N}}orth\textbf{\textit{E}}ast, \textbf{\textit{N}}orth, \textbf{\textit{N}}orth\textbf{\textit{W}}est, \textbf{\textit{W}}est, \textbf{\textit{S}}outh\textbf{\textit{W}}est, \textbf{\textit{S}}outh, \textbf{\textit{S}}outh\textbf{\textit{E}}ast, \textbf{\textit{E}}ast\}.$$

Similar to (Dutta, 1989), in our approach, the elements of the above set are fuzzy numbers defined on the interval [0,1] and described by membership functions. Since direction relations are only approximations, we may apply trapezoidal membership functions to express their vagueness. A (normalized) trapezoidal fuzzy number is represented by the 4-tuple *(a,b,c,d)*, where *a*, *b*, *c*, *d* $\in$ *R*, $\mu_A(a)=\mu_A(d)=0$, and $\mu_A(b)=\mu_A(c)=1$. For a small angle α, *A* corresponds to the following set of fuzzy numbers (graphically represented in Figure 2):

A_1=*NE* $(0^o, 45^o-\alpha, 45^o+\alpha, 90^o)$

A_2=*N* $(45^o, 90^o-\alpha, 90^o+\alpha, 135^o)$

A_3=*NW* $(90^o, 135^o-\alpha, 135^o+\alpha, 180^o)$

A_4=*W* $(135^o, 180^o-\alpha, 180^o+\alpha, 225^o)$

A_5=*SW* $(180^o, 225^o-\alpha, 225^o+\alpha, 270^o)$

A_6=*S* $(225^o, 270^o-\alpha, 270^o+\alpha, 315^o)$

A_7=*SE* $(270^o, 315^o-\alpha, 315^o+\alpha, 360^o)$

A_8=*E* $(315^o, 360^o-\alpha, \alpha, 45^o)$

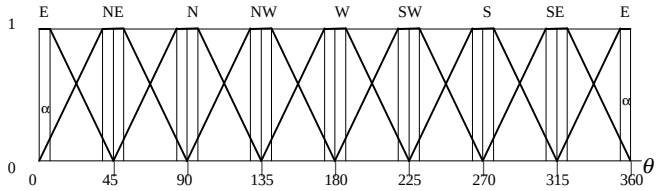


Figure 2 Membership functions for direction relations

Let A_i be a direction relation and θ be a given angle (the angle between the centroids of two database objects). The *direction similarity measure* σ_A between A_i and θ is defined as follows:

$$\sigma_A(A_i,\theta) = \begin{cases} \theta/(i45^o - a) & if\ (i-1)45^0 < \theta < i45^0 - \alpha \\ 1 & if\ i45^0 - \alpha \le \theta \le i45^0 + \alpha \\ ((i+1)45^o - \theta)/(i45^o - \alpha) & if\ i45^0 + \alpha < \theta < (i+1)45^0 \\ 0 & otherwise \end{cases}$$

For instance, in the case of NorthEast:

$$\sigma_A(NE,\theta) = \begin{cases} \theta/(45^o - a) & if\ 0^0 < \theta < 45^0 - \alpha \\ 1 & if\ 45^0 - \alpha \le \theta \le 45^0 + \alpha \\ (90^o - \theta)/(45^o - \alpha) & if\ 45^0 + \alpha < \theta < 90^0 \\ 0 & if\ 90^o \le \theta \le 360^o \end{cases}$$

Similar to topological constraints, a direction constraint C_A is a disjunction (set) of direction relations of A . The similarity between C_A and an angle θ is defined as:

$$\sigma_A(C_A,\theta)=\max(1, \sum_{\forall A_i \in C_A} \sigma_A(A_i,\theta)).$$

Unlike topological relations where a pair of objects may satisfy only one relation, it is possible that two objects satisfy two neighbouring directions at the same time; if, for instance, C_A is $N \vee NE$ and $\theta = 30^o$, then $\sigma_A(C_A,30^o)= 1$. Although neither *North*, nor *NorthEast*, are totally satisfied, their disjunction is. U_A= {*NE, N, NW, W, SW, S, SE, E*} is the universal direction constraint ($\sigma_A(U_A,\theta)=1$ for all θ).

2.3. Distance Constraints

Object centroids are also considered for distances between objects. A distance constraint $D_{di\text{-}dj}$ specifies that the distance between two object centroids must be in the range $[d_i,d_j]$. If $d_i=0$, the constraint corresponds to *closer than* d_j; if $d_j=\infty$, to *farther than* d_i, and if $d_i=d_j$, to *about* d_i. As in the case of direction constraints, we use a trapezoidal membership function to represent such relations. For a small distance δ, $D_{di\text{-}dj}$ corresponds to the fuzzy number $(d_i\text{-}\delta, d_i, d_j, d_j+\delta)$, which is graphically represented in Figure 3. The universal distance constraint is $U_D = D_{0\text{-}\infty}$. Let $D_{di\text{-}dj}$ be a distance constraint and d a given distance. The *distance similarity measure* σ_D between them is defined as follows*:*

$$\sigma_D(D_{di\text{-}dj},d) = \begin{cases} (d - d_i + \delta)/\delta & \text{if } d_i - \delta < d < d_i \\ 1 & \text{if } d_i \le d \le d_j \\ (d_j - d + \delta)/\delta & \text{if } d_j < d < d_j + \delta \\ 0 & \text{otherwise} \end{cases}$$

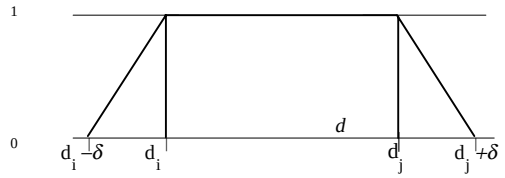


Figure 3 Membership functions for distance relations

The set of spatial relations described in this section constitutes a comprehensive way to express spatial queries. In addition to topology and directions, distances are also taken into account and, to our knowledge, this is the first approach that combines the three types of constraints in similarity retrieval. The parameters τ, α and δ can be tuned to match different application or user needs providing flexibility to the model. However, we do not claim that the above relations and similarity measures exhaust all possibilities. The methods proposed hereafter can be used with alternative definitions.

3. A FRAMEWORK FOR CONFIGURATION SIMILARITY

It is computationally expensive to answer real-time high level queries by processing images representing information at the pixel level. On the other hand, *symbolic representations* facilitate query processing by

explicating the relations of interest. This is in accordance with traditional Information Retrieval techniques, like the *vector processing model* (Salton et al., 1994), which use term vectors instead of the actual text to retrieve document similarity. Let $O=\{o_0,\ldots,o_D\}$ be the set of all objects in the database. Then, according to our definition, an *image* I of a set of objects $O_I \subseteq O$ is a complete graph where each node corresponds to an object in O_I and each arc between o_i and o_j stores: (i) the topological relation between o_i and o_j, (ii) the angle and (iii) the distance between their centroids. Such image representations are called *Attributed Relational Graphs* (ARG) in Computer Vision terminology.

Consider, for example, the physical image shown in the left part of Figure 4, which may be stored in a *Minimum Bounding Rectangle* (MBR) based spatial data structure (e.g., R-trees). Efficient techniques retrieve the topological relations between the actual objects (Papadias and Theodoridis, 1997), while directions and distances can be calculated assuming that the centroid lies in the intersection of the MBR diagonals. The right part of Figure 4 illustrates the equivalent ARG which contains information about the relations of interest (i.e., topology, angle and distance respectively) between o_i and o_j, where i<j (information about the relations between o_j and o_i is straightforward to derive).

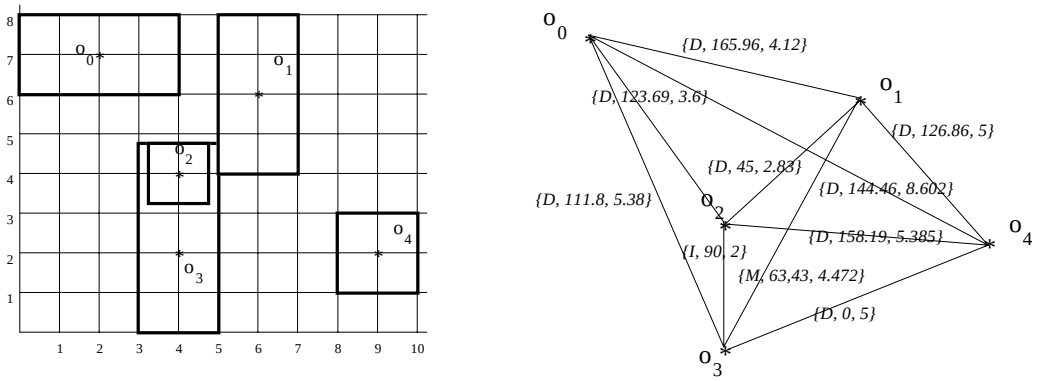


Figure 4 Physical Image and ARG

3.1. Configuration Similarity Queries as Constraint Satisfaction Problems

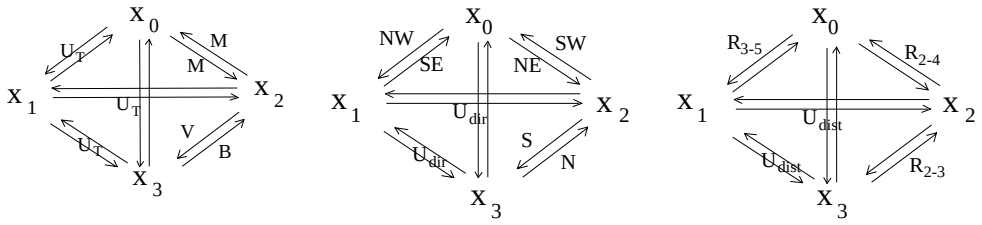
Configuration similarity queries can be formalised as constraint satisfaction problems (CSPs). Formally, a binary CSP is defined as (Mackworth and Freuder, 1985):

- A set of *n* variables, $x_0, x_1, \ldots, x_{n-1}$
- For each variable x_i a finite domain D_i of potential values
- For each pair of variables x_i, x_j a set of binary constraints C_{ij}, where C_{ij} is a subset of $D_i \times D_j$.

Assume, for instance, the query:

> "Retrieve all configurations where there is an object x_0, which is *3-5* distance units *north-west* of an object x_1, and *adjacent* and *north-east* of another x_2 which *contains* x_3 at its *north* side. Furthermore, the distance between the centroids of x_0 and x_2 should be *2-4* distance units, and the distance between the centroids of x_3 and x_2 should be *2-3* units".

This query contains four variables x_0, x_1, x_2, x_3, whose domain is the set of objects O_I in the image *I* to be searched (assuming the image of Figure 4, then $D_i=\{o_0,\ldots,o_4\}$). For each pair of distinct objects there is a topological, direction and distance constraint (some of which may be universal), resulting in a total of *3n(n-1)* binary constraints. Figure 5 illustrates the query constraints in the form of three networks, one for each type of relation. In real applications some additional unary constraints may appear; these may specify object properties (e.g., x_0 is a building), sizes (e.g., x_1 is large), shapes (e.g., x_2 is circular) etc. For generality, we omit such constraints here and deal only with spatial ones. However, unary constraints can be captured by the proposed retrieval framework (Papadias et al., 1998a).



(a) Topological Constraints (b) Direction Constraints (c) Distance Constraints

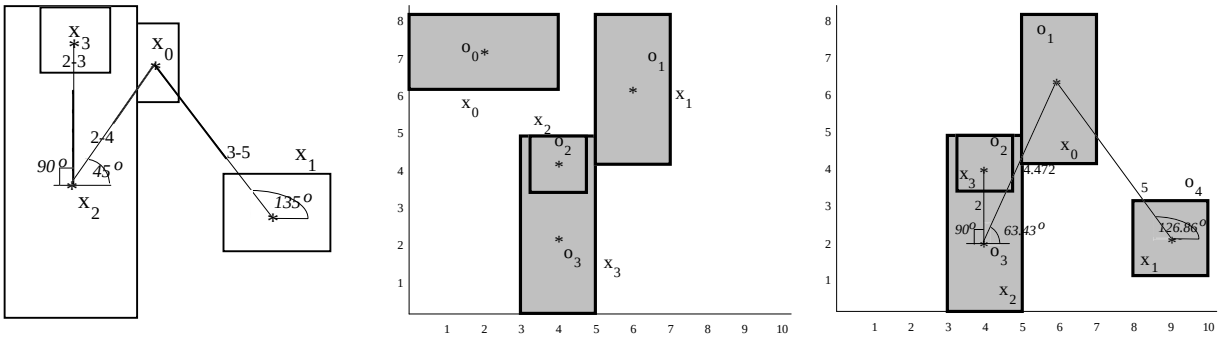
Figure 5 Query constraint networks

The goal is to find instantiations of query variables to image objects such that the input constraints are satisfied to a maximum degree. Consider the binary instantiation $\{x_i \leftarrow o_k, x_j \leftarrow o_l\}$ where variable x_i is instantiated to image object o_k and x_j to o_l. The *degree of satisfaction* of the topological constraint between x_i and x_j equals the topological similarity measure: $\sigma_T(C_T(x_i,x_j), T(o_k,o_l))$. The satisfaction degrees for the other types of constraints are computed accordingly. Assume, for instance, the example query and the binary instantiation $\{x_0 \leftarrow o_0, x_1 \leftarrow o_1\}$ in the image of Figure 4. The degree of satisfaction for the binary constraints that involve x_0 and x_1 are: $\sigma_T(U_T(x_0,x_1), Disjoint(o_0,o_1)) = 1$ (since the topological constraint is unspecified, any such relation fully satisfies it), $\sigma_A(NW(x_0,x_1), 165.96(o_0,o_1)) = 0.351$, and $\sigma_D(D_{3\text{-}5}(x_0,x_1), 4.12(o_0,o_1)) = 1$.

Given the satisfaction degrees of individual constraints, the similarity $\mathbb{S}$ of a complete *solution* $\{x_0 \leftarrow o_p, \ldots, x_{n-1} \leftarrow o_r\}$ can be calculated using several possible metrics. Ruttkay (1994) describes *conjunctive* combination ($\mathbb{S}$ is the minimum degree of satisfaction of individual constraints) and *productive* combination ($\mathbb{S}$ is the product of satisfaction degrees of individual constraints). The problem with conjunctive combination is that it does not distinguish between solutions that contain equally "bad" binary instantiations, while productive combination does not differentiate between instantiations that fully violate some constraint(s). Here we use an *average combination* metric which is the sum of all pairwise similarities divided by the total number of constraints:

$$\mathbb{S} = \frac{\sum_{\forall ij,\, 0\le i,j<n} \sigma_T(C_T(x_i,x_j),T(o_k,o_l)) + \sigma_A(C_A(x_i,x_j),\theta(o_k,o_l)) + \sigma_D(C_D(x_i,x_j),d(o_k,o_l))}{3n(n-1)}, \text{ where } \{x_i \leftarrow o_k,\ x_j \leftarrow o_l\}$$

This avoids the drawbacks of the other metrics (i.e., conjunctive and productive combination) but is more computationally expensive (instantiations cannot be abandoned as early in the search process). Figure 6a illustrates one (of infinite possible) ideal solution of the example query. Due to the query constraints, only the marked angles and distances between the centroids, and the topological relations between x_0 and x_2, and between x_2 and x_3 are important in determining similarity. Figure 6b illustrates the instantiation $\{x_0 \leftarrow o_0, x_1 \leftarrow o_1, x_2 \leftarrow o_2, x_3 \leftarrow o_3\}$ using the image of Figure 4. Although, as we saw before, the satisfaction degrees of constraints between x_0 and x_1 are relatively high (σ_T= 1, σ_A = 0.351, and σ_D=1), the total similarity $\mathbb{S}$ is low because most constraints between other pairs are violated. Figure 6c illustrates another instantiation from the same image, $\{x_0 \leftarrow o_1, x_1 \leftarrow o_4, x_2 \leftarrow o_3, x_3 \leftarrow o_2\}$, which satisfies all topological and most direction and distance constraints.



(a) an ideal configuration (b) a poor solution (c) a good solution

Figure 6 Example instantiations

3.2. Types of Retrieval

There is a trade-off between the level of approximation that we intend to retrieve and the cost of query processing. In general, more approximate solutions involve higher computational cost and vice-versa. We distinguish three types of image retrieval:

- *Hard retrieval*, in which all solutions to be retrieved should totally satisfy all constraints of the query ($\mathbb{S}$=*1)*. The problem is equivalent to a CSP which is NP-Complete (Meseguer, 1989; Grigni et al., 1995). Hard retrieval will not return any solutions if some constraint is not fully satisfied (i.e., if σ < 1), even if there exist configurations that match the query very closely. The following two methods overcome this problem.
- *Soft retrieval*, which finds all solutions that are good on the average (using the *average combination* metric), even if they may totally or partially violate some constraints. This type is considerably more expensive than the first one, because it has to generate more instantiations before it rejects a partial solution (instantiations that violate some constraint are not immediately discarded).
- *Semi-hard retrieval*, which uses the same metric as the previous ones, but excludes solutions that totally violate some constraint (i.e., if σ = 0). The experimental evaluation of Section 6 shows that it

is generally a good trade-off between the first two retrieval types because it reduces execution time, compared to soft retrieval, while missing only few solutions.

Figure 7 illustrates an execution using semi-hard retrieval and a test image with 500 random rectangles. The query (upper right window) contains four variables related with the same spatial constraints as the example query of Figure 5 (*D* stands for distance, *A* for direction and *T* for topological constraint). Notice the existence of several universal constraints, i.e., unconstrained pairs of variables (e.g., (x_0,x_3), (x_1,x_2) etc).

Some solutions and their similarities are given in the lower right window; by clicking on them the user may view the corresponding instantiation in the left window (query variables are in parentheses). Here we illustrate one where x_0 is mapped to object 89 of the image, x_1 to 338 and so on. Its score is 0.876036 (some constraints are partially violated) and its rank 88. Notice that the algorithm required only 594635 instantiations, and this is more or less normal for semi-hard retrieval and queries and images of this size. In order to achieve such efficiency, however, we need some pre-processing techniques described in the next section.

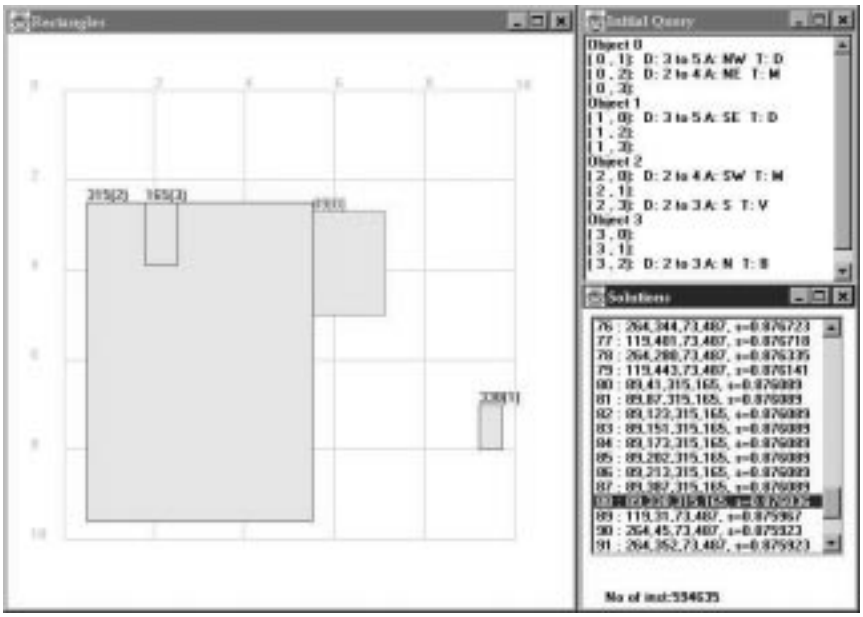


Figure 7 A sample execution

4. QUERY PRE-PROCESSING TECHNIQUES

If *N* is the number of objects in image *I*, and *n* the number of query variables, the total number of possible instantiations is equal to the number of *n*-permutations of the *N* objects: *N!/(N-n)!* (this is equal to $617*10^8$ for the example of Figure 7). In real DBMSs where N>>n, this number is $O(N^n)$, meaning that the retrieval of structural queries can be exponential to the query size. Query processing becomes more expensive if inexact matches are to be retrieved, a situation which arises very often in practical applications. Despite its exponential nature, we have developed two pre-processing methods, *path*

consistency and *variable ordering*, which, when combined with appropriate retrieval algorithms, can effectively solve the problem for real world image and query sizes.

4.1. Path Consistency

The example query contains several implicit constraints, i.e., relations between query variables that are not explicitly stated. For instance, given the distance and direction relations for (x_1,x_0), (x_0,x_2) and (x_2,x_3), it can be inferred that x_1 lies somewhere south-east of the x_3. In addition to their direction, the potential distance between x_1 and x_3 is constrained by the given ranges. In order to explicate such relations, we need *composition tables* that encode rules about the permissible relation for (x_i,x_j), when the constraints for (x_i,x) and (x,x_j) are known. The calculation of the whole set of (explicit and implicit) constraints in a query (i.e., the *query closure*) aims at achieving:

- *efficiency:* tight queries quickly lead to "bad" instantiations of variables that effectively prune the search space;
- *early detection of inconsistent queries*: in hard retrieval, inconsistent queries (e.g., "find all images where x_i is *in* x_j which is *in* x_k and x_k *meets* x_i") always have no solutions and therefore, there is no need to be executed; furthermore, they provide feedback for potential user errors.

Two composition tables are required: one for topological and one for combined distance and direction relations. Table 1 illustrates the composition rules for topological relations (Egenhofer, 1991)[4]. Since, in the context of this paper, topological relations are defined on extended objects, they are completely independent from the distance and direction ones (which are defined on centroids). No conclusion[5] can be drawn about the direction and the distance between the centroids of two objects given their topological relation (and vice versa), unless we take advantage of domain knowledge (e.g., buildings 3-5 kms apart are *disjoint*). We intend to provide a general framework for the problem, not tied to any specific application, thus we assume independence of topological relations.

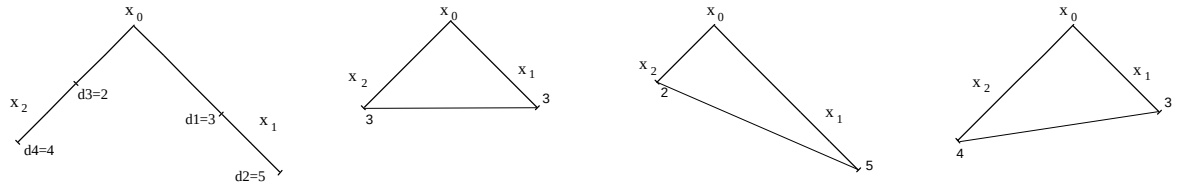
	Disjoint(x$_i$,x$_j$)	**Meet(x$_i$,x$_j$)**	**Equal(x$_i$,x$_j$)**	**Inside(x$_i$,x$_j$)**	**coverBy(x$_i$,x$_j$)**	**Contain(x$_i$,x$_j$)**	**coVer(x$_i$,x$_j$)**	**Overlap(x$_i$,x$_j$)**
Disjoint(x$_i$,x)	U_T	D∨M∨I∨B∨O	D	D∨M∨I∨B∨O	D∨M∨I∨B∨O	D	D	D∨M∨I∨B∨O
Meet(x$_i$,x)	D∨M∨C∨V∨O	D∨M∨E∨B∨V∨O	M	I∨B∨O	M∨I∨B∨O	D	D∨M	D∨M∨I∨B∨O
Equal(x$_i$,x)	D	M	E	I	B	C	V	O
Inside(x$_i$,x)	D	D	I	I	I	U_T	D∨M∨I∨B∨O	D∨M∨I∨B∨O
coverBy(x$_i$,x)	D	D∨M	B	I	I∨B	D∨M∨C∨V∨O	D∨M∨E∨B∨V∨O	D∨M∨I∨B∨O
Contain(x$_i$,x)	D∨M∨C∨V∨O	C∨V∨O	C	E∨I∨B∨C∨V∨O	C∨V∨O	C	C	C∨V∨O
coVer(x$_i$,x)	D∨M∨C∨V∨O	M∨C∨V∨O	V	I∨B∨O	E∨B∨V∨O	C	C∨B	C∨V∨O
Overlap(x$_i$,x)	D∨M∨C∨V∨O	D∨M∨C∨V∨O	O	I∨B∨O	I∨B∨O	D∨M∨C∨V∨O	D∨M∨C∨V∨O	U_T

Table 1 Composition table for topological relations

[4] Composition of topological relations has been applied to detect inconsistencies in spatial databases (Smith and Park, 1992; Egenhofer and Sharma, 1993) and optimize query processing in (Papadias et al., 1995).

[5] This is not true for projection-based definitions of directions relations; sometimes topological information can be conveyed from directions and vice-versa (Sharma, 1996; Papadias and Theodoridis 1997).

On the other hand, direction and distance relations are interrelated. Consider again the example query where the explicit relation between x_1 and x_0 is $SE \wedge D_{3\text{-}5}$ (x_0 is *NW* of x_1), and between x_0 and x_2 is $NE \wedge D_{2\text{-}4}$ (Figure 8a). From this information, we calculate the implicit direction and distance relations between x_1 and x_2. Regarding directions, x_1 could be exactly *east* (Figure 8b) or *E-SE* (Figure 8c) of x_2. If the distance between x_1 and x_0 were unconstrained and approached $+\infty$, the direction between x_1 and x_2 would approach *SE*. Similarly, x_1 could also be *E-NE* of x_2 (Figure 8d). Given the distance constraints, the minimum distance between x_1 and x_2 is $\sqrt{2^2+3^2}$ (when both x_1 and x_2 are at the closest positions to x_0), while the maximum distance is $\sqrt{4^2+5^2}$. The general composition rule for $SE \wedge D_{d1\text{-}d2}$ and $NE \wedge D_{d3\text{-}d4}$ is $(NE \vee E \vee SE) \wedge D_{m\text{-}M}$ where $m=\sqrt{d_1^2+d_3^2}$ and $M=\sqrt{d_2^2+d_4^2}$ (NE, E, SE are the only relations with potentially non-zero memberships).



(a) explicit relations (b) direction E (c) direction E-SE (d) direction E-NE

Figure 8 Example of distance-direction composition

Table 2 illustrates the complete inferences about direction and distance relations (the example of Figure 8 corresponds to the entry in the last row and fourth column). The table covers the most general case in the sense that it contains all relations that may have non-zero membership. However, depending on the given ranges, further pruning of the direction relations may be possible and is performed by our pre-processing methods. In the example of Figure 8, if we knew that d_1 is greater than d_4, we would infer that x_1 can only be SE or E of x_2 (not NE).

In this pre-processing phase, a path consistency algorithm (Mackworth and Freuder, 1985; Papadias and Egenhofer, 1997) explicates constraints (according to Tables 1 and 2) and creates the query closure. The composition constraint is computed by: (i) generating the cross products of the relation sets that form the constraints for (x_i,x) and (x,x_j); (ii) composing each ordered pair by looking up the results in the table, and (iii) taking the union of the resulting sets. Constraints are refined when the composition constraint contains a proper subset of relations of the original one. If some relation is left unspecified, the composition constraint contains the corresponding universal relation. When the intersection of the composition and original constraint is empty there is an inconsistency, in which case the system informs the user.

Figure 9 illustrates the transitive closure of the example query where new constraints have been generated by path consistency (disjunctions are denoted by strings of relations). Notice that in this case, universal relations have been replaced by relatively tight constraints. Going back to the example in the

	$NW\wedge D_{d3\text{-}d4}(x,x_j)$	$N\wedge D_{d3\text{-}d4}(x,x_j)$	$NE\wedge D_{d3\text{-}d4}(x,x_j)$	$W\wedge D_{d3\text{-}d4}(x,x_j)$	$E\wedge D_{d3\text{-}d4}(x,x_j)$	$SW\wedge D_{d3\text{-}d4}(x,x_j)$	$S\wedge D_{d3\text{-}d4}(x,x_j)$	$SE\wedge D_{d3\text{-}d4}(x,x_j)$
$NW\wedge D_{d1\text{-}d2}(x_i,x)$	$NW\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(NW\vee N)\wedge D^{*}$	$(NW\vee NE\vee N)\wedge D^{**}$	$(NW\vee W)\wedge D^{*}$	$(NW\vee N\vee NE\vee E)\wedge D^{***}$	$(NW\vee N\vee NE)\wedge D^{**}$	$(NW\vee W\vee SW\vee S)\wedge D^{***}$	$(NW\vee SE)\wedge D^{****}$
$N\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee N)\wedge D^{*}$	$N\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(N\vee NE)\wedge D^{*}$	$(N\vee NW\vee W)\wedge D^{**}$	$(N\vee NE\vee E)\wedge D^{**}$	$(N\vee NW\vee W\vee SW)\wedge D^{***}$	$(N\vee S)\wedge D^{****}$	$(N\vee NE\vee E\vee SE)\wedge D^{***}$
$NE\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee NE\vee N)\wedge D^{**}$	$(N\vee NE)\wedge D^{*}$	$NE\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(NE\vee N\vee NW\vee W)\wedge D^{***}$	$(NE\vee E)\wedge D^{*}$	$(NE\vee SW)\wedge D^{****}$	$(NE\vee E\vee SE\vee S)\wedge D^{***}$	$(NE\vee E\vee SE)\wedge D^{**}$
$W\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee W)\wedge D^{*}$	$(N\vee NW\vee W)\wedge D^{**}$	$(NE\vee N\vee NW\vee W)\wedge D^{***}$	$W\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(W\vee E)\wedge D^{****}$	$(W\vee SW)\wedge D^{*}$	$(W\vee SW\vee S)\wedge D^{**}$	$(W\vee SW\vee S\vee SE)\wedge D^{***}$
$E\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee N\vee NE\vee E)\wedge D^{***}$	$(N\vee NE\vee E)\wedge D^{**}$	$(NE\vee E)\wedge D^{*}$	$(W\vee E)\wedge D^{****}$	$E\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(E\vee SE\vee S\vee SW)\wedge D^{***}$	$(E\vee SE\vee S)\wedge D^{**}$	$(E\vee SE)\wedge D^{*}$
$SW\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee N\vee NE)\wedge D^{**}$	$(N\vee NW\vee W\vee SW)\wedge D^{***}$	$(NE\vee SW)\wedge D^{****}$	$(W\vee SW)\wedge D^{*}$	$(E\vee SE\vee S\vee SW)\wedge D^{***}$	$SW\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(SW\vee S)\wedge D^{*}$	$(SW\vee S\vee SE)\wedge D^{**}$
$S\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee W\vee SW\vee S)\wedge D^{***}$	$(N\vee S)\wedge D^{****}$	$(NE\vee E\vee SE\vee S)\wedge D^{***}$	$(W\vee SW\vee S)\wedge D^{**}$	$(E\vee SE\vee S)\wedge D^{**}$	$(SW\vee S)\wedge D^{*}$	$S\wedge D_{(d1+d3)\text{-}(d2+d4)}$	$(S\vee SE)\wedge D^{*}$
$SE\wedge D_{d1\text{-}d2}(x_i,x)$	$(NW\vee SE)\wedge D^{****}$	$(N\vee NE\vee E\vee SE)\wedge D^{***}$	$(NE\vee E\vee SE)\wedge D^{**}$	$(W\vee SW\vee S\vee E)\wedge D^{***}$	$(E\vee SE)\wedge D^{*}$	$(SW\vee S\vee SE)\wedge D^{**}$	$(S\vee SE)\wedge D^{*}$	$SE\wedge D_{(d1+d3)\text{-}(d2+d4)}$

$D^{*} = D_{m\text{-}M}$, where $m = \sqrt{d_1^2 + d_3^2 + \sqrt{2}d_1d_3}$ and $M = \sqrt{d_2^2 + d_4^2 + \sqrt{2}d_2d_4}$

$D^{**} = D_{m\text{-}M}$, where $m = \sqrt{d_1^2 + d_3^2}$ and $M = \sqrt{d_2^2 + d_4^2}$

$D^{***} = D_{m\text{-}M}$, where $M = \max(\sqrt{d_1^2 + d_3^2 - \sqrt{2}d_1d_3}, \sqrt{d_1^2 + d_4^2 - \sqrt{2}d_1d_4}, \sqrt{d_2^2 + d_3^2 - \sqrt{2}d_2d_3}, \sqrt{d_2^2 + d_4^2 - \sqrt{2}d_2d_4})$

$$\text{and } m = \begin{cases} \min(\phi, d_1\sqrt{2}/2), if\ d_1\sqrt{2}/2 \in [d_3, d_4] \\ \min(\phi, d_2\sqrt{2}/2), if\ d_2\sqrt{2}/2 \in [d_3, d_4] \\ \min(\phi, d_3\sqrt{2}/2), if\ d_3\sqrt{2}/2 \in [d_1, d_2] \\ \min(\phi, d_4\sqrt{2}/2), if\ d_4\sqrt{2}/2 \in [d_1, d_2] \\ \phi, otherwise \end{cases}$$

where $\phi = \min(\sqrt{d_1^2 + d_3^2 - \sqrt{2}d_1d_3}, \sqrt{d_1^2 + d_4^2 - \sqrt{2}d_1d_4}, \sqrt{d_2^2 + d_3^2 - \sqrt{2}d_2d_3}, \sqrt{d_2^2 + d_4^2 - \sqrt{2}d_2d_4})$

$D^{****} = D_{m\text{-}M}$, where $M = \max(|d_2\text{-}d_3|, |d_1\text{-}d_4|)$

$$\text{and } m = \begin{cases} 0, & if\ d_3 \le d_2 \quad and \quad d_1 \le d_4 \\ \min(|d_2 - d_3|, |d_1 - d_4|), & otherwise \end{cases}$$

Table 2 Composition table for distances and directions

beginning of this section, observe that the possible directions between x_1 and x_3 are E∨SE∨S, while their distance lies between 3.36796 and 7.94725.

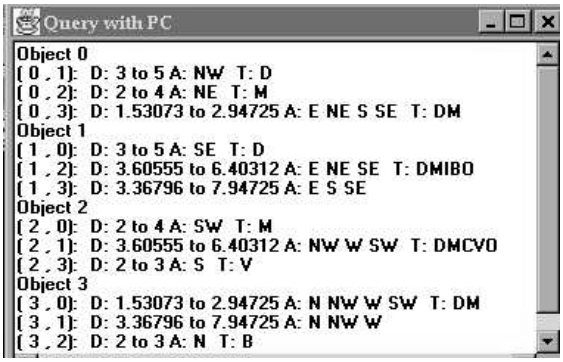
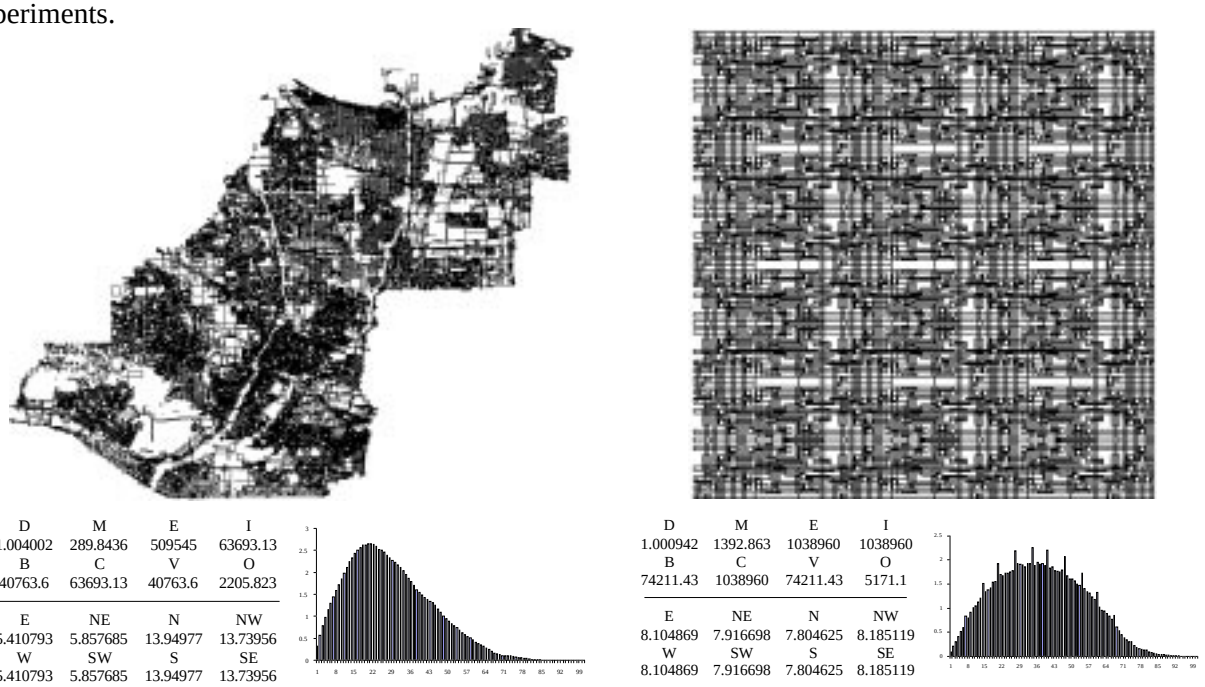


Figure 9 Closed query

4.2. Variable Ordering

The second phase of pre-processing determines the order in which variables get instantiated, a factor that significantly affects the search space. In general, a good ordering is one where the most constrained variables are instantiated first, because bad instantiations are detected and abandoned early in the search (Dechter and Meiri, 1994). We use a *weight W(R,I)* to express the rareness of a topological or direction relation *R* in a particular image *I*. In order to calculate *W(R,I)* we perform an exhaustive search in each image *I* and count the pairs of objects N_R that satisfy relation *R* in *I*. *W(R,I)* is considered to be equal to $N(N-1)/N_R$ (where *N(N-1)* is the number of distinct object pairs). According to this rationale, relations that occur rarely get high weights because they have high discriminative value (this is similar to *inverse term frequency* used in information retrieval techniques). For instance, if a query specifies that two variables are *equal*, then these two variables should be instantiated first in order to prune the search space as early as possible. On the other hand, *disjoint* has a very small weight since, for normal data density, it is satisfied by more than 95% of the object pairs; therefore, a *disjoint* constraint is not significant. Figure 10 illustrates the weights of topological and direction relations as derived in two images used in our experiments.



(a)Weights and distance distributions - roads of Long Beach county (b)Weights and distance distributions-VLSI Image

Figure 10 Distributions of weights and distances

Given the pre-computed weights of relations, weights for constraints (i.e., disjunctions of relations of the same type) are calculated by the following equation, which captures the property that the weight of a constraint is smaller than the weight of any (tighter) constraint with a proper subset of its relations:

$$\frac{1}{W(C,I)} = \sum_{\forall Rj \in C} \frac{1}{W(R_j,I)}$$

For instance, the weight of a constraint $C_A = NE$ in the image of Figure 4a would be 5.857, while the weight of another $C_{A'} = N$ would be 13.949. The weight of a (less restrictive) constraint $NE \vee N$ would be: 5.857*13.949/(5.857+13.949) = 4.124. We also computed the distribution of distances normalized by the maximum distance among all pairs of objects (Figure 10). Weights of ranges were calculated from these distributions using the above equation since a range can be thought of as a disjunction of precise distances (the longer the range the smaller its weight). Image meta-data (weights, distance distribution) are calculated only once and stored with each image. When an image *I* is searched for a particular configuration expressed by *Q*, its associated meta-data are retrieved and each query variable is assigned a weight which is equal to the sum of weights of constraints in which it participates. Instantiation order of variables in *Q* for retrieval from *I* is then determined according to the variable weights: the "heaviest" variable first and the least constrained last.

If the example query is to be processed using the above input images, then x_2 should be the variable to be instantiated first because it is involved in the most restrictive constraints (e.g., *covers* x_3 and *meets* x_0). The second variable should be x_3 followed by x_0. The experiments described in Section 6 demonstrate the improvement achieved by the above pre-processing techniques.

5. RETRIEVAL ALGORITHMS

This section describes how several constraint satisfaction algorithms can be modified for configuration similarity retrieval. Search is performed solely using the ARG which is maintained in main memory. The *K* best complete mappings (*K* is user-defined), according to the chosen retrieval type (hard, semi-hard, soft) are retrieved and shown to the user. We developed four retrieval algorithms: three of them are based on backtracking, while the fourth one is based on forward checking.

5.1 Backtracking-based algorithms

The first algorithm is a non-recursive variation of chronological backtracking (BT). After the ARG has been constructed, every query variable is instantiated to an image object according to the order determined during query pre-processing. When a variable is instantiated to some object, this object is "*locked*", i.e., it is removed from the domain of current and future variables and cannot be mapped to another query object. The new similarity is calculated by adding to the previous one the satisfaction degrees of the constraints that relate the new object with already instantiated objects. Depending on the new similarity and the type of retrieval used, the mapping proceeds forward (to the next variable) or backward. The algorithm proceeds forward if the instantiations so far constitute a *partial solution*; i.e., if all similarity degrees are 1 (for hard retrieval), or if the current similarity can exceed the target similarity of the K^{th} solution (for soft retrieval). The condition for semi-hard is the same as for soft, provided that no constraints have been totally violated. When the algorithm goes backward, another mapping for the

same variable is chosen and the previous object is unlocked. If the variable domain is empty, the algorithm proceeds another step back and re-instantiates the previous query variable after releasing the locks of all objects locked by subsequent variables.

A simplified version of the algorithm is illustrated in Figure 11. *Current_instantiations* is an 1D array of *n* elements that holds the current values of variables (*current_instantiations[i]* holds the current value of x_i). *S[i]* holds the current similarity at the instantiation level *i* (variables up to the i^{th} one have been instantiated). *Solutions* is a *K*x*n* array that holds the *K* instantiations that have the highest similarity. *Target* is the similarity of the K^{th} solution; an instantiation will be included in the solutions only if its similarity is greater than *target*. When a variable is to be instantiated, BT chooses a value from its domain and calculates the *new_similarity* produced by the new instantiation and the already instantiated variables. If the new instantiation produces a partial solution, the algorithm proceeds forward, or outputs a solution if there are no other un-instantiated variables. Otherwise, it chooses a new value for the current variable. For simplicity, the locking/unlocking of values is omitted from the code.

```
BT (Query q, Image I, int K )
Preprocess(Q) /* compute query closure, retrieve I and metadata, and determine variable ordering (most constrained first) */
FOR j := 0 TO n-1 DO domain[j] = O_I /*initialize all domains to O_I */
S = target = 0;
i = 0; /* index to the current variable */
WHILE (TRUE) {
    new_similarity := 0;
    new_value := chooseNextValue(domain[i][i]);
    IF new_value = NULL  /* empty domain */
        THEN
            IF i=0 THEN RETURN; /* end of domain for first variable - end of the algorithm */
            ELSE i:=i-1; CONTINUE;  /*Backtrack*/
        ELSE /* non-empty domain */
            current_instantiations[i] := new_value; /*store instantiation*/
            FOR j=0 to i-1  /*calculate the new similarity produced by the instantiation of current variable */
                new_similarity = new_similarity +σ(C_ji,R(current_instantiations[j], current_instantiations[i]));
            S[i] = S[i-1] + new_similarity;
            IF S[i] can exceed target  /*instantiated variables 0,..,i constitute a partial solution (depending on retrieval type)*/
                THEN IF i < n-1 /* intermediate variable instantiated */
                    THEN   i := i+1; /* successful instantiation: go forward */
                    ELSE /*last variable instantiated*/
                        store(current_instantiations, solutions); /* solution stored */
                        target = solutions[K];
}
```

Figure 11 Backtracking for configuration similarity retrieval

The second retrieval algorithm is based on Backjumping (BJ) (Dechter, 1990). Assume, for example a query where the instantiation order is x_0, x_1, x_2, x_3, x_4, and that there does not exist a value in the domain of x_4 which is consistent with the instantiations of the previous variables. Furthermore, this inconsistency is solely due to the constraint between and x_4 and x_1 (e.g., a restrictive constraint such as *covers*). Backtracking to x_3 or x_2 will not solve the problem (the constraints between x_4 and x_3 or x_2 may be non-restrictive or even universal). Backjumping, on the other hand, will re-instantiate x_1, that caused the

problem, thus reducing the number of consistency checks. In order to do this, a pointer has to be kept for each variable to the last variable that caused an inconsistency. The forward move is the same as in backtracking, checking the constraints of the current variable with respect to previously instantiated ones.

In the above example, BJ after re-instantiating x_1, would move forward, re-instantiate x_2 and so on. Assume that x_2 and x_3 are related by a restrictive constraint satisfied only by a few object pairs. In such a case, finding a good instantiation pair for x_2 and x_3 may require a significant amount of search which is basically redundant since such a pair was already found before the deadlock at x_4 occurred. Unlike BJ, Dynamic Backtracking (DBT) (Ginsberg, 1993), after re-instantiating x_1, would keep the existing instantiations of x_2 and x_3, and directly attempt to find a consistent value for x_4. This means that the instantiation order is changed dynamically, i.e., from x_0, x_1, x_2, x_3, x_4 it becomes x_0, x_2, x_3, x_1, x_4, so DBT can be thought of BJ with *dynamic variable ordering* (Bachus and Van Run, 1995).

5.2 Forward checking-based algorithm

One of the most effective constraint satisfaction algorithms is *forward checking* (FC) (Haralick and Elliot, 1980; Bachus and Grove, 1995) which has been shown to outperform the rest for a wide range of problems involving "crisp" constraints. FC must be modified for configuration similarity queries in order to handle *soft* constraint processing. The adjusted version works as follows: when a variable x_i is assigned a value o_k, the domain of each *future* (un-instantiated) variable x_j is pruned according to o_k and the constraint C_{ij}, for all j>i. That is, all values o_l that produce similarities that cannot exceed the *target* are removed from the domain of x_j. Consequently, when we reach instantiation level *i*, the values of variables $x_0,\ldots,x_{i-1}$ will constitute a partial solution, and the domains of future variables will contain only values that may lead to a (complete) solution given the instantiations so far.

The procedure of pruning the domains of the future variables is called *check forward*. If, after a check forward, the whole domain of a future variable is eliminated, the algorithm discards the current variable's value, and restores the values of future variables, which were eliminated due to the current instantiation. When the domain of the current variable is exhausted, the algorithm *backtracks* to the previous one and assigns a new value to it. FC outputs a solution whenever the last variable is given a value, and terminates when it backtracks from the first variable.

In order to keep track of the allowable values for each variable at every instantiation level, FC uses a $n_X n_X N$ *domain table*. Each element of *domain[i][j]* is an array of *N* values that x_j can take at different levels. Before FC starts, *domain[0][j]* is initialized to O_I for all variables. When x_0 is assigned a value o_p, *domain[1][j]* is computed for each remaining x_j, by including only values $o_l \in$ *domain[0][j]* that can exceed *target*. In general, if o_k is the current value of x_i, *domain[i+1][j]* is the subset of *domain[i][j]* which is valid w.r.t. C_{ij} and o_k. In this way, at each instantiation level the *domain[i][j]* of x_j continuously shrinks; when we reach level j, x_j gets instantiated from *domain[j][j]* which contains only values compatible with the instantiations of previous variables. If a value of x_i results in the domain of some x_j to

become empty, a new value is chosen and *domain[i+1][j]* is re-initialized to *domain[i][j]*. Figure 12 contains the pseudo-code of FC.

```
FC (Query q, Image I, int K )
Preprocess(Q) /* compute query closure, retrieve I and metadata, and determine variable ordering (most constrained first) */
FOR j = 0 TO n-1 DO domain[0,j] = O_I /*initialize all domains to O_I */
S = target = 0;
i = 0; /* index to the current variable */
WHILE (TRUE) {
    new_value := chooseNextValue(domain[i][i]);
    IF new_value = NULL /* empty domain */
        THEN IF i=0  /* end of domain for first variable */
                THEN RETURN; /* end of algorithm */
                ELSE i:=i-1; CONTINUE;  /*Backtrack*/
        ELSE /*non- empty domain */
                current_instantiations[i] := new_value; /*store instantiation*/
                IF i = n-1 /*last variable instantiated*/
                    THEN
                        store (current_instantiations, solutions);
                        target=solutions[K];
                    ELSE /* intermediate variable instantiated */
                        IF check_forward(i) /* successful instantiation*/
                            THEN i := i+1; /* successful instantiation: go forward */
 }
BOOLEAN check_forward(int i)
  FOR j = i+1 TO n-1 DO /*for all uninstantiated variables*/
      domain[i+1][j]= domain[i][j];
      FOR all objects o_l ∈ domain[i+1][j]
          FOR k=0 to i-1
              new_similarity = new_similarity +σ(C_ki,R(current_instantiations[k], o_l));
          S[i] = S[i-1] + new_similarity;
          IF S[i] cannot exceed target /* depending on the type of retrieval*/
                  THEN domain[i+1][j]= domain[i+1][j]-{o_l};  /* remove o_l from the domain */
      IF domain[i+1][j]=∅ THEN RETURN FALSE; /* a future domain becomes empty (the current value of x_i is illegal) */
  RETURN TRUE;
```

Figure 12 Forward checking for configuration similarity retrieval

Figure 13 illustrates the execution of semi-hard FC using the example query and the image of Figure 4. Initially all *domains[0][j]* are O_I for each x_j. As discussed in 4.2, the order of variables after pre-processing becomes: x_2, x_3, x_0, x_1. When x_2 is instantiated to o_0, *domain[1][3]* becomes empty because there is no object that satisfies $D_{2\text{-}3} \wedge S \wedge covers(x_2,x_3)$ for $x_2 \leftarrow o_0$ (the constraints between all pairs of variables are illustrated in Figure 9). Thus, the instantiation $x_2 \leftarrow o_0$ is abandoned and another value is chosen for x_2 (for demonstration we illustrate what would be the domains of other variables at this point). Instantiations $x_2 \leftarrow o_1$ and $x_2 \leftarrow o_2$ are unsuccessful for the same reason (e.g., o_1 and o_2 do not *cover* any other object). When x_2 is instantiated to o_3 the domains of future variables become: *domain[1][3]*=$\{o_2\}$, *domain[1][0]*=$\{o_1\}$ and *domain[1][1]*=$\{o_4\}$. Following these instantiations, as shown in Figure 13, we reach the solution of Figure 6c. The algorithm will then attempt to re-instantiate $x_2 \leftarrow o_4$ (there are no further values in the domains of the other variables) because o_4 is the only value not yet assigned to x_2. This instantiation will cause the domain of x_3 to become empty and FC will terminate.

instantiation level i	*domain[i][2]*-x_2	*domain[i][3]*-x_3	*domain[i][0]*-x_0	*domain[i][1]*-x_1	Illustration
initialization	$\{o_0,o_1,o_2,o_3,o_4\}$	$\{o_0,o_1,o_2,o_3,o_4\}$	$\{o_0,o_1,o_2,o_3,o_4\}$	$\{o_0,o_1,o_2,o_3,o_4\}$	
i=1 $\{x_2 \leftarrow o_0\}$	$\{o_1,o_2,o_3,o_4\}$ (in addition to o_0, x_2 can be instantiated to any other object of O_1)	$\varnothing$ (there is no object that satisfies $D_{2\text{-}3} \wedge S \wedge covers(o_0,x_3)$)	$\varnothing$ (there is no object that satisfies $D_{2\text{-}4} \wedge NE \wedge meet(o_0,x_0)$)	$\{o_1,o_2,o_3\}$ (these objects satisfy $D_{3.6\text{-}6.4} \wedge \{NW,W,SW\} \wedge \{D, M,C,V,O\}$ (o_0,x_1))	
i=1 $\{x_2 \leftarrow o_3\}$	$\{o_4\}$	$\{o_2\}$ (the only object that satisfies $D_{2\text{-}3} \wedge S \wedge$ $covers(o_3,o_2)$)	$\{o_1\}$ (the only object that satisfies $D_{2\text{-}4} \wedge SW$ $\wedge meet(o_3,o_1)$)	$\{o_4\}$ (the only object that satisfies $D_{3.6\text{-}6.4} \wedge$ $\{NW,W,SW\} \wedge \{D,$ $M,C,V,O\}$ (o_0,o_4))	
i=2 $\{x_2 \leftarrow o_3, x_3 \leftarrow o_2\}$	$\{o_4\}$	-	$\{o_1\}$	$\{o_4\}$	
i=3 $\{x_2 \leftarrow o_3, x_3 \leftarrow o_2,$ $x_0 \leftarrow o_1\}$	$\{o_4\}$	-	-	$\{o_4\}$	
i=4 $\{x_2 \leftarrow o_3, x_3 \leftarrow o_2,$ $x_0 \leftarrow o_1, x_1 \leftarrow o_4\}$	$\{o_4\}$	-	-	-	

Figure 13 Demonstration of forward checking

6. EXPERIMENTAL EVALUATION

In order to test the performance of the algorithms we used real geographic and VLSI data sets of various sizes. In particular, we constructed images of 100, 200, 300, 400 and 500 objects using portions of the map and the VLSI image in Figure 10 (a total of ten images). The parameters of Section 2 for spatial relations were set to: $\tau=0.33$, $\alpha=5$ and $\delta=0$, while K=100. Since actual queries may vary significantly depending on the domain, we constructed an artificial set of 70 queries each consisting of 3 to 9 variables (10 queries of 3 variables, 10 of 4, …, 10 of 9 variables). Query tightness varies from complete queries created using a query-by-sketch language[6] to very loose queries involving only a few non-restrictive constraints. The implementation was done using Java Symantec JIT compiler and the experiments were run on several Pentium PCs 133MHz with 64M Ram.

[6] The user draws the configuration on a sketch-board and all spatial constraints are automatically generated from the object locations (see Section 7).

We executed all 70 queries, for the ten images using the four algorithms (BT, BJ, DBT and FC) with and without pre-processing, for the three retrieval types (hard, soft, semi-hard) (i.e. a total of 16800 runs). Each execution was allowed 10 minutes to complete; after this period it was terminated. Figure 14 illustrates the average times (in seconds) for each algorithm and retrieval type combination, for queries with four variables and images of 300 objects. FC has the best performance for all types of retrieval (the other combinations of query/image sizes yield very similar relative performance). The trade-off is that it needs $O(n^2N)$ space for keeping the domain table, as opposed to $O(nN)$ of the other algorithms (to store the domain of each variable).

Among the variations of backtracking, BJ is the best alternative for the current application. Because of the small number of query variables and the structure of the problem, DBT is slightly more expensive due to the overhead of dynamic variable ordering. DBT, in general, performs well when the problem can be split in independent clusters of variables. However, path consistency generates constraints between most query variables; therefore, the existence of independent clusters of variables is highly unlikely.

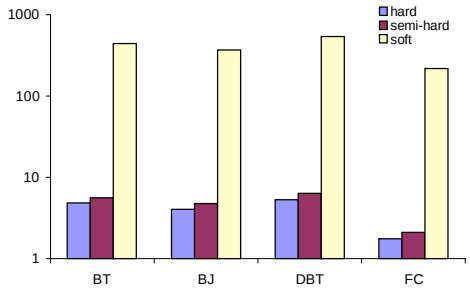


Figure 14 Comparison of algorithms for configuration similarity

Another interesting observation from the above graph concerns the different retrieval types. Soft retrieval is 1-2 orders of magnitude slower than semi-hard retrieval and prohibitively expensive for real-time applications. On the other hand, semi-hard retrieval is only about 10% slower than hard retrieval, while in 90% of the cases it retrieved the same solutions as soft (solutions that are good on the average while totally violating some constraint are rare, especially for large images). In general, semi-hard seems to be a very good trade-off for applications involving images of 100 or more objects and require approximate retrieval.

Figure 15 illustrates the effect of pre-processing on the performance of FC. Each chart shows the response time (in milliseconds) as a function of the number of query variables (3,..,9) and the image size (100,…,500). Soft retrieval did not terminate successfully for most queries involving more than four variables. The experiments illustrate that pre-processing speeds up query processing more than an order of magnitude for all types of satisfiability. The same is true for the rest of the algorithms. For most queries, semi-hard FC with pre-processing will find solutions in less than 10 seconds even if image sizes reach 500 objects.

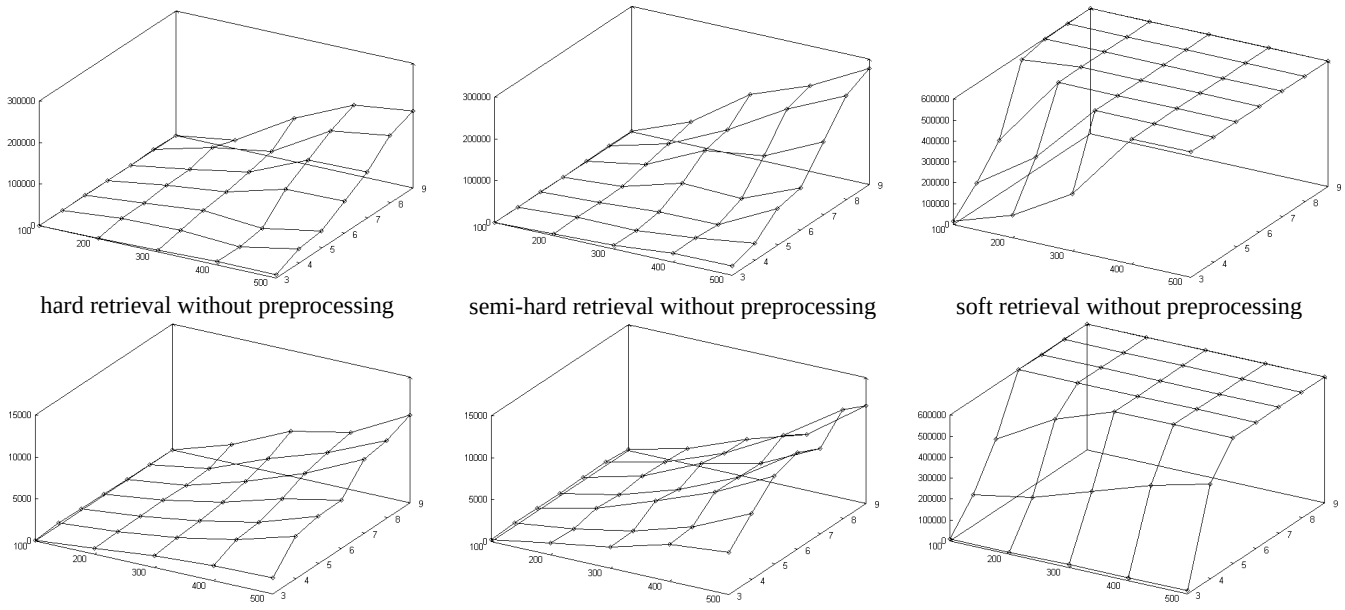


Figure 15 FC performance

In actual applications, much larger images could be effectively processed since queries often involve some other features that prune the search space. GIS queries, for instance, usually include properties of objects (e.g., "find all maps where there is a river crossing a large city" etc) that restrict variable domains to a small percentage of the image size. Papadias et al., (1998b) show how R*-trees (Beckmann et al., 1990) can be used to facilitate retrieval using very large images (where the size does not permit the whole process to take place in main memory). Depending on the query properties, images of up to 50,000 objects can be effectively processed.

7. DISCUSSION

Development of efficient retrieval methods will significantly aid the spatial database community to exploit the large, and steadily increasing, amount of related data available in the form of satellite images, topographic maps etc. This paper provides a framework for configuration similarity retrieval that accommodates all major types of spatial constraints and handles the inherent vagueness of spatial queries. We consider three cases of retrieval (hard, semi-hard and soft) and apply several constraint satisfaction algorithms for query processing. Pre-processing techniques, which involve computation of the transitive closure of the query and reordering of variables, significantly improve performance. Our framework could be used locally, within a single query processor, or on the World Wide Web, as a part of spatial search engines.

Currently, we work on query languages for configuration similarity retrieval. The expression of multiple spatial constraints among numerous variables using a verbal query language (e.g., SQL) is complicated and counter-intuitive. Pictorial languages, where the user draws the query on a sketch-board

(Smith and Chang, 1996), are more friendly and help avoid inconsistencies. The problem with such languages, however, is that they are too restrictive. By drawing a pair of objects on the board, the user specifies the exact distance, angle and topological relation, in addition to some object properties (e.g., size), although this may have not been the original intention (for instance, the query example used throughout the paper cannot be drawn without making unnecessary commitments about the unspecified relations).

An ideal approach should combine advantages of both worlds, namely, the flexibility of verbal and the usability of pictorial languages. We have developed a query-by-sketch language for configuration similarity retrieval which includes several of the above features, and can be executed from the WWW using any standard browser. Figure 16 illustrates the initial page where the user can draw a configuration to be matched in the database. The language allows the enabling/disabling of topological and direction constraints through "tick boxes", and the manual specification of distances through a "pop-up" menu (as in Figure 16). The algorithm and the type of retrieval can also be selected (semi-hard forward checking is the default option). The "threshold" specifies the minimum similarity that a configuration should have in order to be retrieved.

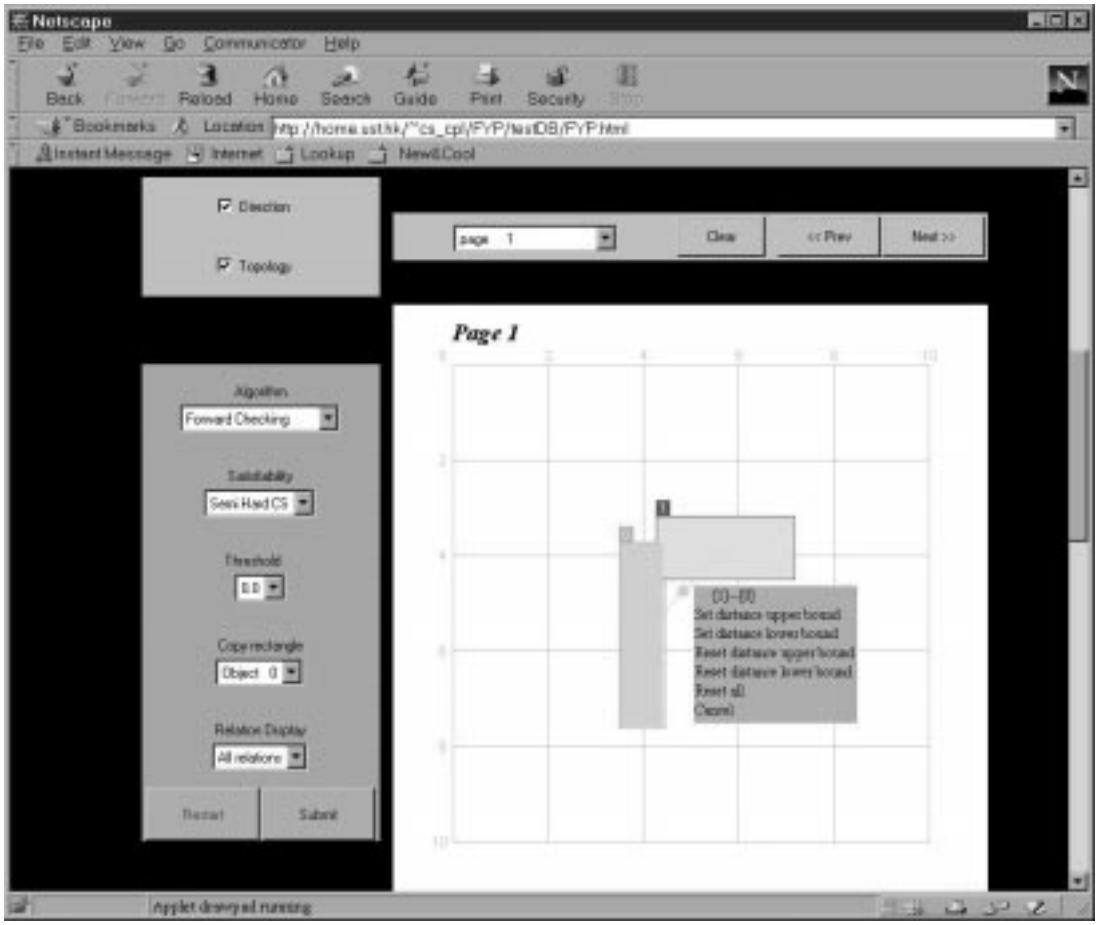


Figure 16 Query-by-draw

Consider that a user wants to retrieve all configurations containing three variables x_0, x_1 and x_2, where x_0 and x_1 are related as in Figure 16, and x_2 is *SE* of x_0, without specifying the relative position between x_1 and x_2. Query-by-sketch allows the expression of such queries using *multiple pages*. The second page (Figure 17) illustrates the constraint between x_0 and x_2. A copy of x_0 is created using the "copy rectangle" option. Topological relations and distances are disabled in the second page, while kept in the first one. In this way complicated queries, involving many pages and several variable copies can be intuitively expressed.

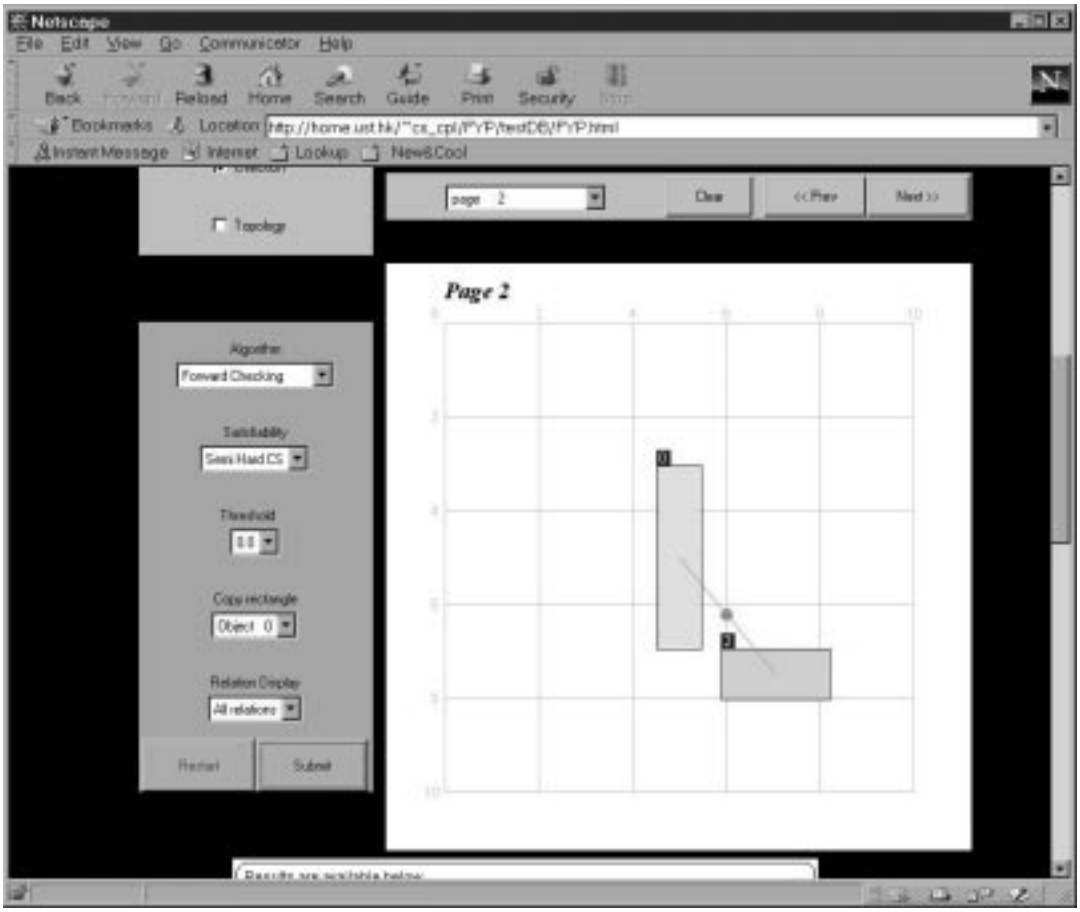


Figure 17 Second page in query-by-draw

Future work includes comparison (or even combination) of the proposed algorithms with alternative ones that have worked well in other domains. We intend to examine other search algorithms such as, *hill climbing* (Minton et al., 1992), *tabu search* (Glover, 1990) and *simulated annealing* (Johnson et al, 1991). Furthermore, our retrieval algorithms could be enhanced to include additional features, such as rotation and mirror transformations. Configuration similarity could be also combined with visual matching algorithms to form more complicated queries (e.g., "find all images where there is a *brown* building near a *circular* lake"). Our framework could easily model such queries using unary constraints. In order to process visual features, however, appropriate matching techniques (e.g., Jagadish, 1991) are required.

Acknowledgements: Parts of the paper are based on (Papadias et al., 1998a) and (Papadias et al., 1998c). We would like to thank Nikos Mamoulis, Dimitris Meretakis and Chung Po Loi for helping with the implementation. This work was supported by grants DAG96/97.EG36 and HKUST6151/98E from Hong Kong Research Grant Council.

REFERENCES

Agouris, P., Stefanidis, A., Carswell, J. (1998) "Digital Image Retrieval Using Shape-based Queries". Proceedings of the 8th International Symposium on Spatial Data Handling (SDH).

Bach, J.R., Paul, S., Jain, R. (1993) "A Visual Information Management System for the Interactive Retrieval of Faces". IEEE TKDE, 5(4), 619-627.

Bacchus, F., Grove, A. (1995) "On the Forward Checking Algorithm", International Conference on Principles and Practice of Constraint Programming.

Bacchus, F., Van Run, P. (1995) "Dynamic Variable Ordering for CSPs", International Conference on Principles and Practice of Constraint Programming.

Ballard, D., Brown, C. (1984) *"Computer Vision"*. Prentice Hall.

Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B. (1990) "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles". ACM SIGMOD.

Bruns,T., Egenhofer, M. (1996) "Similarity of Spatial Scenes". 7th Symposium on Spatial Data Handling.

Burrough, P., Frank, A. (1996) *"Geographic Objects with Indetermined Boundaries"*. Taylor-Francis.

Chang, S.K., Shi, Q.Y. Yan C.W. (1987) "Iconic Indexing by 2-D String". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no 3, 413-428.

Chu, W.W., Ieong, I., Taira, R. (1994) "A semantic modeling approach to image similarity retrieval by content". VLDB Journal, 3, 445-477.

Dechter, R. (1990) "Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition". Artificial Intelligence 41, 273-312.

Dechter, R., Meiri, I. (1994) "Experimental evaluation of preprocessing algorithms for constraint satisfaction problems". Artificial Intelligence 68, 211-241.

Dubois, D., Fargier, H. and Prade, H. (1993) "Propagation and satisfaction of flexible constraints". Yager, R.R. and Zadeh, L. A. (eds.): Fuzzy Sets, Neural Networks and Soft Computing, Kluwer Academic.

Dutta, S. (1989) "Qualitative Spatial Reasoning: A Semi-Qualitative Approach using Fuzzy Logic". In Buchmann, A., Günther, O., Smith, T., Wang, T., (eds.) Proceedings of the 1st Symposium on the Design and Implementation of Large Spatial Databases (SSD). Springer Verlag LNCS.

Egenhofer, M. (1991) "Reasoning about Binary Topological Relations". In Günther, O. and Schek, H.J. (eds.) Proceedings of the 2nd Symposium on the Design and Implementation of Large Spatial Databases (SSD). Springer Verlag LNCS.

Egenhofer, M., Franzosa R. (1991) “Point Set Topological Relations”. International Journal of Geographic Information Systems, 5, 161-174.

Egenhofer, M., Al-Taha, K. (1992) “Reasoning about Gradual Changes of Topological Relations”. International Conference GIS - From Space to Territory, Springer, LNCS.

Egenhofer, M., Sharma, J. (1993) “Assessing the Consistency of Complete and Incomplete Topological Information”. Geographical Systems, Vol. 1, No 1, 47-68.

Egenhofer, M. (1997) “Query Processing in Spatial-Query-by-Sketch”. Journal of Visual Languages and Computing, Vol. 8, 403-424.

Frank, A. U. (1996) “Qualitative Spatial Reasoning: Cardinal Directions as an Example”. International Journal of Geographic Information Systems, Vol. 10 (3), 269-290.

Ginsberg, M. (1993) “Dynamic Backtracking”. Journal of Artificial Intelligence Research, 1, 25-46.

Glover, F. (1990) “Tabu search: A tutorial”. Interfaces, 20, 74-94.

Grigni, M., Papadias, D., Papadimitriou, C. (1995) “Topological Inference”, Proceedings of the 14th International Joint Conference of Artificial Intelligence.

Gudivada, V., Raghavan, V. (1995) “Design and evaluation of algorithms for image retrieval by spatial similarity”. ACM Transactions on Information Systems, 13(1):115-144.

Haralick, R. M., Elliott, G. L. (1980) “Increasing Tree Search Efficiency for Constraint Satisfaction Problems”. Artificial Intelligence 14:263-313.

Hernandez, D. (1994) "*Qualitative Representation of Spatial Knowledge*", Springer Verlag LNAI.

Jagadish H. V. (1991). “A retrieval technique for similar shape”. Proceeding of the ACM SIGMOD International Conference on Management of Data.

Johnson, D.S., Aragon, C.R., McGeogh, L.A., Shevon, C. (1991). “Optimization by simulated annealing: An experimental evaluation”. Operations Research, 39, 378-406.

Lee S-Y., Hsu F-J (1992) “Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation”. Pattern Recognition, 25(3):305-318.

Mackworth, A, Freuder, E. (1985) “The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems”. Artificial Intelligence, 25, pp. 65-74.

Mark, D., Egenhofer, M. (1994) “Modeling Spatial Relations Between Lines and Regions: Combining Formal Mathematical Models and Human Subjects Testing”. Cartography and Geographical Information Systems 21 (3): 195-212.

Meseguer, P. (1989) “Constraint satisfaction problems: an overview” AICOM, 2 (1), 3-17.

Minton, S., Johnston, M.D., Philips, A.B., Laird, P. (1992) “Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems”. Artificial Intelligence, 58, pp. 161-205.

Nabil, M., Ngu, A., Shepherd, J. (1996) “Picture similarity retrieval using 2d projection interval representation”. IEEE TKDE, 8(4), pp. 533-539.

Orphanoudakis, S., Chronaki, C., Kostomanolakis, S (1994) “I2C: A System for the Indexing, Storage and Retrieval of Medical Images by Content”. Technical Report 113, Institute of Computer Science, Foundation for Research and Technology - Heraklion, Greece.

Papadias, D., Sellis, T. (1995) "A Pictorial Query-By-Example Language". Journal of Visual Languages and Computing, Special Issue on Visual Query Systems, 6(1), pp. 53-72.

Papadias, D., Theodoridis, Y., Sellis, T., Egenhofer, M. (1995) "Topological Relations in the World of Minimum Bounding Rectangles: a Study with R-trees". Proceedings of the ACM Conference on the Modeling of Data (SIGMOD), San Jose, CA, ACM Press.

Papadias, D., Theodoridis, Y. (1997) “Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures”. International Journal of Geographic Information Science, 11(2), pp. 111-138.

Papadias, D., Egenhofer, M. (1997) “Algorithms for Hierarchical Spatial Reasoning”. Geoinformatica, Vol. 1 (3), pp. 251-273.

Papadias, D., Arkoumanis, D., Karacapilidis, N. (1998a) "On The Retrieval of Similar Configurations". Proceedings of the 8th International Symposium on Spatial Data Handling (SDH).

Papadias, D., Mamoulis, N., Delis, B. (1998b) "Algorithms for Querying by Spatial Structure". Proceedings of 24th International Conference on VLDB. New York.

Papadias, D., Mamoulis, N., Meretakis, D. (1998c) "Image Similarity Retrieval by Spatial Constraints". Proceedings of 7th International Conference on Information and Knowledge Management (ACM-CIKM). Washington DC.

Pappis, C., Karacapilidis, N. (1993) “A comparative assessment of measures of similarity of fuzzy values”. Fuzzy Sets and Systems, Vol. 56, pp. 171-174.

Petrakis, E., Faloutsos, C. (1997) "Similarity Searching in Medical Image Databases". IEEE TKDE, 9 (3) 435-447.

Rabitti, F., Savino, P., (1992) “An Information Retrieval Approach for Image Databases”. Proceedings of the 18th International Conference on VLDB.

Ruttkay, Z. (1994) "Fuzzy Constraint Satisfaction". IEEE Conference on Fuzzy Sets and Systems.

Salton, G., Allan, J., Buckley, C. (1994) “Automatic Structuring and Retrieval of Large Text Files”. Communications of the ACM, 37(2): 97-108.

Sharma, J. (1996) "Integrated Spatial Reasoning in GIS: Combining Topology and Direction". Ph.D. Thesis, Department of Spatial Information Science and Engineering, University of Maine, Orono, ME.

Sistla, A., Yu, C. Y., Haddad, R. (1995) “Similarity based Retrieval of Pictures using indices on Spatial Relationships”. Proceedings of the 21st International Conference on VLDB.

Smith, J. R., Chang, S-F., (1996) “Searching for images and videos on the World-Wide Web”. Technical Report CU/CTR 459-96-25, Columbia University.

Smith, T., Park, K., (1992) "Algebraic Approach to Spatial Reasoning". International Journal of Geographic Information Systems, Vol. 6, No. 3, pp. 177-192.

Tagare, H.D., Jafee, C.C., Duncan, J.S. (1992) “Arrangement: A Spatial Relation for Describing and Comparing Part Embeddings”. Proceedings of 11th International Conference On Pattern Recognition.

Topaloglou, T. (1996) “Spatial Databases with Partial Information: Representation and Reasoning” Ph.D. Thesis, Dept. of Computer Science, University of Toronto, Canada, 1996.

Zadeh, L.A. (1965) “Fuzzy Sets”. Information and Control, Vol. 8, pp. 338-353.

Zimmermann, H.J. (1991) *“Fuzzy Set Theory and its Applications”*. Boston, MA, Kluwer.