A Provably Efficient Computational Model For Approximate Spatiotemporal Retrieval

Delis Vasilis[1,2], Makris Christos[1] and Sioutas Spiros[1]
[1]Computer Engineering and Informatics Dept. University of Patras, Greece
[2]Computer Technology Institute Patras, Greece
tel: +3061 960332 fax: +3061 960322
delis@cti.gr, {makri, sioutas}@ceid.upatras.gr

Abstract
The paper is concerned with the effective and efficient processing of spatiotemporal selection queries under varying degrees of approximation. Such queries may employ operators like *overlaps*, *north*, *during*, etc., and their result is a set of entities standing approximately in some spatiotemporal relation θ with respect to a query object X. The contribution of our work is twofold: i) First we describe a mathematical framework for representing multidimensional relations at varying granularity levels, modelling relation approximation through the concept of relation convexity. ii) We subsequently exploit the framework for developing approximate spatiotemporal retrieval mechanisms, employing a set of existing as well as new main memory and secondary memory data structures that achieve either optimal or the best known performance in terms of time and space complexity, for both static and dynamic problems.

1 OVERVIEW

The handling of spatiotemporal information is an increasingly evident demand for modern DBMSs, initially motivated by applications like GIS, CAD, etc. and soon expanded in areas such as robotics, medical imaging, multimedia applications, etc. [GG98]. Efficient query processing for this type of data is a feature of primary importance.

Spatiotemporal databases are collections of entities which have either spatial attributes (e.g. geographic databases) or temporal attributes (e.g. medical databases) or combinations thereof (e.g. multimedia databases). Spatial attributes can be viewed as 0D, 1D, 2D or 3D positions in a "space", either the physical one (e.g. map objects) or an artificial one such as a computer screen (e.g. multimedia objects). Temporal attributes capture the temporal existence of entities and in the general case can be represented as time points or time intervals.

In order to support querying of such attributes, a DBMS must provide appropriate operators, usually in the form of binary spatial and temporal predicates (or relations), such as *contains*, *west*, *near*, *after*, etc. (e.g. "find all buldings *northeast* of the park built *during* 1970-1980"). A main category of queries of this sort is the class of *spatiotemporal selection* queries which ask for all entities standing in some spatiotemporal relation θ to a query object X. The processing of such selection queries gives rise to three main requirements, which are addressed in this work:

- a formal framework for the effective representation of spatiotemporal relations
- handling of approximate matches
- appropriate data structures for efficient query processing

The study of binary spatial and temporal relations have been traditionally among the research interests of the DB and AI communities, respectively (see [PS94, Ege97, Kau, All83]). This interaction has yielded an abundance of models, which however suffer from a particular limitation: they allow for the representation of fixed sets of relations, thus restricting the potential range of applications. To our knowledge, no existing model provides varying granularity levels in the description of relations. Moreover, there is no universally accepted uniform spatiotemporal model.

Responding to these requirements, we propose a multidimensional relation framework that can be tuned in various relation detail levels, which is based on 1D partitions called *resolution schemes*. The set of feasible relations at every resolution level are represented as binary strings which are characterised by an inherent poset structure, called *conceptual neighbourhood*, which provides certain desirable properties. An early informal description of the relation framework has been presented in [PMD98] and applied for 2D configuration retrieval (queries that ask for a set of objects that satisfy a particular spatial configuration), whereas a rigorous formulation can be found in [DH99]. Section 2 in this paper borrows from the latter the concepts that pertain to spatiotemporal selection queries and provides the corresponding relation background.

Unlike traditional selection queries where the output consists of a set of exact matches, in spatiotemporal queries we are equally interested in approximate matches as well, because i) inherently, answers closely similar to an exact match are not equally important to dissimilar ones, and ii) there are no universally accepted semantics for some spatiotemporal predicates like *north-*

east (see Fig. 1). Instead of dealing with approximate matches for queries employing exact relations, one can equivalently accept exact matches for queries on approximate relations. A trivial way to deal with such uncertainty is to treat an ambiguous or approximate relation as a disjunction of all potential exact candidate relations. The drawback however is that such a query is implemented as a set of subqueries, each corresponding to a candidate relation, whose outcome must be eventually combined (at the cost of extra processing overhead). In Section 3 we present an alternative elegant way to define approximate relations, free of the undesirable query processing overhead, based on the concept of *relation convexity*. We also map an ND selection query to a N'D ($N'=2N$) *range query* problem, which asks for all N'D points contained in a N'D range (or *window*). The constituent 1D ranges can be semi-infinite or finite intervals. Unlike efforts in the relevant literature that try to reduce dimensionality, this transformation allows us to exploit the rich set of available solutions for the relatively well studied range query problem.

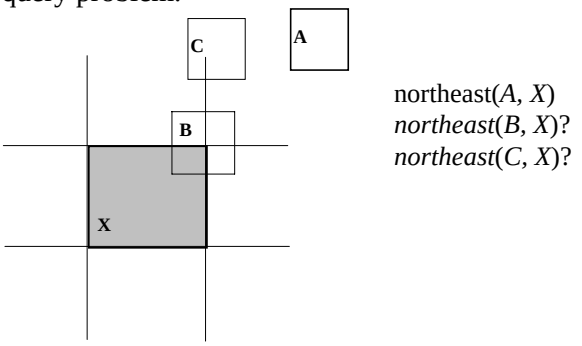

Figure 1 Example of Approximate Spatial Matches

Section 4 presents a suite of main memory and secondary memory data structures (some already published and some new) that answer multidimensional range queries in provably optimal or the best known asymptotic time. Due to space limitations, time and space complexities are sketch proven but can be found in full in [DMS99]. The significance of our results is obvious for 1D (temporal indexing applications), 2D (GIS and spatial DB applications), 3D (viewed as either volume handling applications, or multimedia applications -2D snapshots over time-).

The majority of the related work makes use of spatial indexing schemes, most often R-trees and their variants [Gut84, SRF87, BKSS90], to answer a subset of spatial queries in 2D (usually topological or directional) [PT97, TPSS98]. Lately R-trees have been also utilised for higher dimensionalities [TVS96]. In contrast, employing our 2D relation framework we can answer a richer set of queries based on user-tuned relations (including topological, directional and approximate distance relations).

The wide adoption of secondary memory indexing schemes, especially the R-tree family, for this type of problems, is mainly attributed to their satisfactory average performance in terms of page accesses. However, this does not undermine approaches towards main memory structures, for several factors: i) theoretical average performance estimation for the R-tree family has been particularly evasive (see [TS96] for some late results) while worst case performance can be very bad ii) the assumption that most searches are I/O-bound than CPU-bound (which justifies the choice of secondary memory structures) is not always true in spatial data management [Gae95, HS95] and iii) the availability of increasingly larger memories at lower costs justifies the use of main memory structures even for the processing of large datasets.

Nowadays, for up to three dimensions there are available secondary memory structures, discussed in Section 4, that exhibit worst case performance analogous to that of main memory ones, i.e. query time of the form $O(log_B n+t/B)$ and space usage close to linear ($O(n/B)$), for range queries (where n is the number of stored items, B the page size and t the output size).

However, for higher dimensionalities and cases where good worst-case performance is critical, approaches utilising main memory structures must be considered as possible alternatives to secondary memory indexing schemes. For spatiotemporal selection queries the above practically mean that: i) 1D relation selection queries (transformed to 2D range queries) can be very efficiently answered using secondary memory structures, ii) for 2D, 3D and generally ND selection queries (equivalently, 4D, 6D and 2ND range queries) there exists a choice of classic multidimensional indexing schemes plus a family of very efficient main memory structures.

2 Relation Framework

The two probably most popular formal models for the representation of temporal and spatial relations are Allen's interval algebra [All83] and Egenhofer's intersection model [EH91], respectively. The former proposes a complete set of pair-wise disjoint relations between temporal intervals while the latter suggests that the topological relations between spatial objects can be determined considering the emptiness or not of the pair-wise intersections among the objects' boundaries and interiors.

We argue however that there is no a priori complete

set for spatiotemporal relations for any class of entities. Rather, each time the application at hand indicates the appropriate type and resolution of relations to be considered. Egenhofer's model provides the assumption that relationships in space (and time) can be qualitatively described by the emptiness or not of intersections (coincidence of spatial or temporal occupancy) while Allen's relations indirectly suggest that different amounts of detail can be captured in the description of a relation between intervals, depending on the definition of appropriate 1D regions of interest.

Inspired by these ideas, we propose a *multiresolution* relation framework, considering initially relations in 1D. Given a *reference* interval $[a,b]$, we can identify five potential regions of interest: 1.$(-\infty,a)$ 2.$[a,a]$ 3.(a,b) 4.$[b,b]$ 5.$(b,+\infty)$. Given an arbitrary interval $[c,d]$ (which we call *primary*), its relation to $[a,b]$ can be uniquely determined by the five intersections of $[c,d]$ with each of the above five regions, the emptiness of which are modelled by five binary variables t_1, t_2, t_3, t_4, t_5, ("0" corresponds to an empty intersection while "1" corresponds to a non-empty one). For example, R_{11000} corresponds to the *left-meet* relation (—_). The different relations that can be defined in this way coincide with Allen's [All83] interval relations (*before, after, overlap, during, starts, finishes*, their six converse relations and *equals*).

The choice of regions of interest can be arbitrary and in fact the particular choice (thus, the number of relation bits) determines the resolution level in the description of relations. Let $Y=<X_1, X_2, \ldots, X_n>$ be a partition of $(-\infty,\infty)$. Given an interval X, let $X.l$ and $X.r$ represent its left and right endpoint (regardless of closure at endpoints), respectively (in case X has zero length, $X.l=X.r$). Also, let $seq(Y)=<x_1, x_2, \ldots, x_m>$ be the ordered sequence of the elements of $\{X_1.l, X_1.r, X_2.l, X_2.r, \ldots, X_n.l, X_n.r\}-\{-\infty,\infty\}$ (e.g. $seq((-\infty,a), [a,a], (a,b), [b,b], (b,+\infty)) = <a, b>$).

Definition 1. Assume a reference interval $[a,b]$. A *resolution scheme* (or simply *scheme*) $Y(a,b) = <X_1, X_2, \ldots, X_n>$ is a partition of $(-\infty,\infty)$, where $seq(Y(a,b)) = <x_1, x_2, \ldots, x_m>$. A relation R over Y is a binary string $t_1t_2\ldots t_n$, $t_i \in \{0,1\}$, t_i="1" iff $X_i \cap [x,y] \neq \emptyset$ and t_i="0" otherwise, $i=1..n$, complying with the following constraints: i) it contains exactly one substring of consecutive "1"s and ii) there is at least one t_i="1", in a position i such that X_i is a non-zero length interval. Given a primary interval $[x,y]$ we use the notation $R_{01100}([x,y],[a,b])$ to indicate that the relation between $[x,y]$ and $[a,b]$ is represented by the string 01100. For any reasonably practical scheme a and b must be included in $seq(Y(a,b))$ i.e. $\exists k\ \exists p,\ k<p\leq m$: $x_k=a$, $x_p=b$.

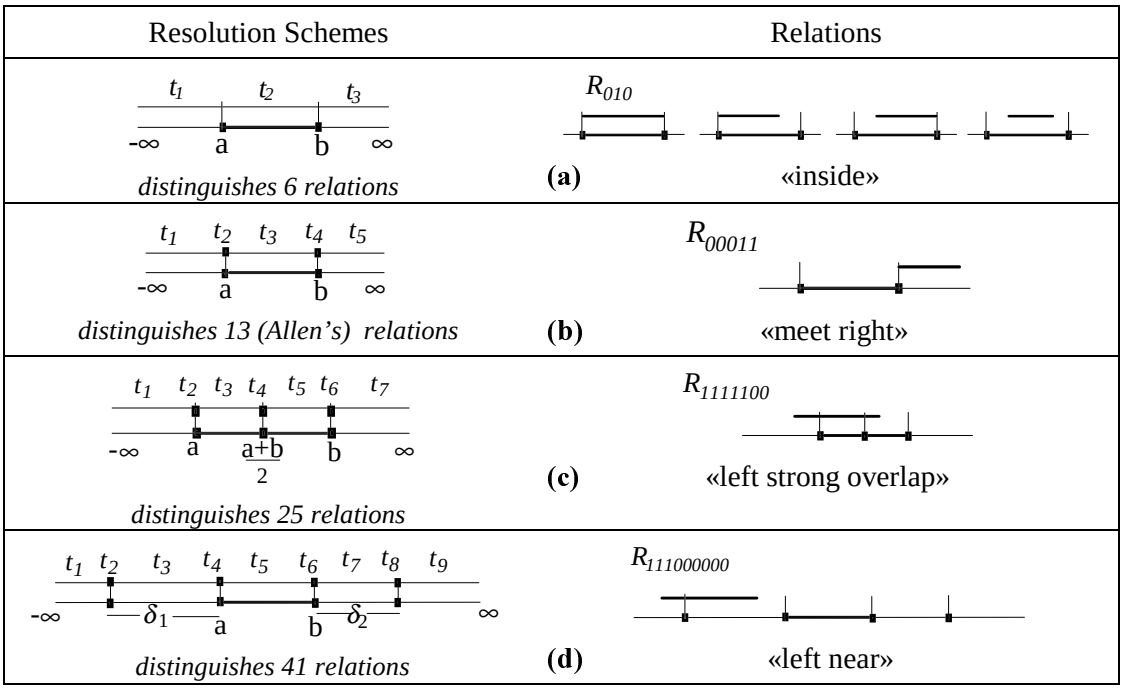

Figure 2 Example Resolution Schemes and Corresponding Relations

A few indicative resolution schemes, some example relation strings and the corresponding interval configurations are given in Figure 2 and described below:

- $(-\infty,a)$ $[a,b]$ $(b,+\infty)$: a coarse scheme where meet (at endpoints) relations can not be distinguished. Figure 2.a portrays different interval relationships that

correspond to the same example relation string. This example shows that the endpoints of regions of interest need not necessarily constitute zero-length regions by themselves, unless meeting or not at the endpoints should discriminate two relations. This is not the case in this scheme.

- $(-\infty,a)$ $[a,a]$ $(a,(a+b)/2)$ $[(a+b)/2,(a+b)/2]$ $((a+b)/2, b)$ $[b,b]$ $(b,+\infty)$: a scheme which refines overlap relations (Figure 2.c).
- $(-\infty,a-\delta_1)$ $[a-\delta_1,a-\delta_1]$ $(a-\delta_1,a)$ $[a,a]$ (a,b) $[b,b]$ $(b,b+\delta_2)$ $[b+\delta_2,b+\delta_2]$ $(b+\delta_2,+\infty)$: a distance enhanced scheme, where *near* is defined as being in a distance of up to δ and *far* otherwise (Figure 1.e). Left and right distances need not necessarily be symmetric.

The intuition behind the above is the following: In order to distinguish several *left* or *right* relations with respect to a reference interval, one can establish appropriate regions of interest by defining points to the left and to the right of [a,b]. An analogous definition of regions of interest by establishing appropriate points in [a,b] allows the description of several *overlap* relations (e.g. *strong overlap* as at least 50% coverage of [a,b] vs. *weak overlap*, defined as less than 50% coverage). In general, such points can de defined at either absolute distances (e.g. a-d) or at relative to the length of [a,b] distances (e.g. $b+\lambda(b-\alpha)$).

The binary string encoding of relations is particularly useful for a certain kind of relation reasoning, involving efficient calculation of relation similarity and relation neighbourhood computation (see [DH99]). For our purposes however any notational convention would equally do. The distinguishable relations at a particular scheme are called *primitive* relations. In general, if n is the number of bits used by the resolution scheme, the number of distinguishable relations (equivalently, the number of all binary strings satisfying the previous constraints) is $n(n+1)/2-k$, where k is the number of bits assigned to zero-length regions of interest (e.g. for the scheme in Fig. 2.b n=5, k=2). The set of primitive relations at every scheme is inherently permeated by a partial order, defined as:

$$X \leq Y \text{ iff}$$

$$rm(X) \leq rm(Y) \wedge lm(X) \leq lm(Y)$$

where $X=x_1x_2\ldots x_n, Y=y_1y_2\ldots y_n$, $x_i, y_i \in \{0,1\}$, are relation strings, $rm(X)$, $lm(X)$ return the position of the rightmost and leftmost "1", respectively, in the binary string of a relation X.

The corresponding Hasse diagram (which is in fact a distributive lattice) is called *conceptual neighbourhood graph*. It has the property that similar relations are closer to each other than non-similar ones, a fact which we will exploit in the next section in order to define approximate relations. Figure 3 presents such an example graph for Allen's relations (the least and greatest lattice elements are the leftmost and rightmost relations in the figure, respectively). This particular graph was first proposed by Freksa [Fre91] as a means to capture similarity among interval relations (two relations are linked through an edge if they can be transformed to each other by a continuous minimal deformation of the primary interval). In this work however we have formalised and generalised the definition.

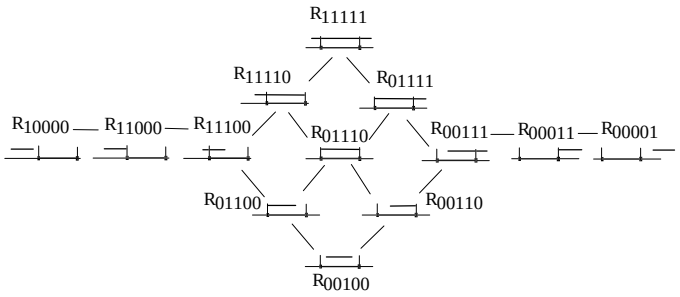

Figure 3 Neighbourhood Graph for Scheme of Fig. 2b

So far we have established a flexible framework for the definition of 1D relations at varying detail levels. This can be easily extended to multiple dimensions for the definition of *projection-based* relations among ND rectangles. Such rectangular approximations of real objects (Minimum Bounding Rectangles -MBRs- being the most popular representatives for 2D) are often utilised for multidimensional querying as the facilitate the so called two-step query processing: i) at a *filter step* non qualifying candidates are very fast eliminated, resulting in a set of potential answers to the query, then ii) at a *refinement step* the exact objects of the answer set are examined to detect possible false hits.

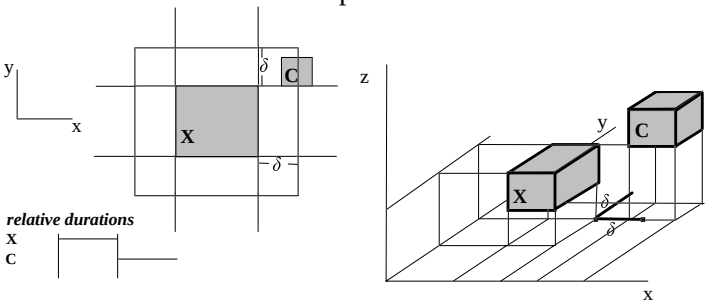

Figure 4 Configurations for $R_{000000111\text{-}000001100\text{-}00011}(C,X)$

The relation between two such ND objects is determined by the combination of N 1D relations, each corresponding to the relation between the object projections for each of the dimensions. Thus an ND relation can be naturally defined as a N-tuple of 1D relations, e.g. $R_{11000\text{-}11100} = \langle R_{11000}, R_{11100} \rangle$. So, depending on the particular application, relations at each

of the dimensions need not necessarily be defined at the same resolution scheme. Assuming the distance-enhanced scheme for the *x* and *y* axes, and Allen's scheme for the *z* and *time* axes, Figure 4 illustrates an example of a 3D relation, interpreted either as a relation between volumes or as a relation between planar objects with temporal existence (e.g. multimedia objects), by giving the $R_{x\text{-}y\text{-}z}$, $R_{x\text{-}y\text{-}time}$ representation.

There are $S_1S_2\ldots S_n$ distinct ND relations, where S_i is the number of distinguishable relations at the *i*-th dimension. All of the properties of relations in 1D can be directly extended in ND. For example, ND relations are partially ordered where the ND partial order is the logical combination of *N* 1D partial orders like the one defined above, one for each of the dimensions. Also, ND conceptual neighbourhood graphs are fractal graphs, i.e. graphs whose nodes are graphs whose nodes are graphs, etc., for *N* levels of nesting (see [DH99]). In the next section we introduce a concept which allows us to define approximate relations.

3 Approximate Relation Querying

We have already stressed in the introductory section the need for a formal definition of approximate relations, as an effective means to deal with approximate query matches. Typical selection queries ask for all entities standing in some spatiotemporal relation θ to a query object *X*, where θ can in general be a set of ND primitive relations for a particular value of *N*, semantically corresponding to their disjunction. In the two extreme cases, θ is a singleton (corresponding to an exact query relation), or the full set of primitive relations, called the *universal relation* (indicating complete lack of knowledge for a particular relationship). In between, θ being a proper subset of the set of all primitive relations introduces some degree of uncertainty in the description of the requested relationship.

Such arbitrary relation subsets are bound to processing overhead as they can be only implemented as combinations of several distinct queries, each for every primitive relation in the set. In the sequel we will restrict the form of such sets in a way that does not introduce extra processing cost.

Definition 2. A *convex relation* (the term was coined by Ligozat [Lig91] in a different context though) is an interval in the relation lattice at a particular resolution scheme, i.e. a relation of the form [*R*, *S*] where *R*, *S* are primitive relations (thus including exact relations as a special case).

Therefore, an ND relation is convex if its 1D constituent relations are convex. An example of a 1D convex relation in $[R_{11100}, R_{01110}]=\{R_{11100}, R_{01100}, R_{11110}, R_{01110}\}$ (see Fig. 3). Intuitively, convex relations capture a continuous uncertainty (contain all intermediate relations between *R*, *S* as potential candidates) in our knowledge of a particular relationship. They are more reasonable approximations than arbitrary disjunctive relations since in practical cases they are likely to arise due to inexact/ambiguous observations (e.g. when asking for directions, it is more likely to get an answer "the post office is two or three or four blocks away", than "the post office is two blocks or ten blocks away").

Viewing each 1D interval as a point in the $x<y$ half-plane, a 1D relation is represented by a 0D, 1D or 2D region in the half-plane (see Fig. 5). Convex relations have an elegant geometrical characterisation, which is a convex point-set (a point-set is convex if the points of a line segment that connects any two points belong to the set), as expressed by the following theorem.

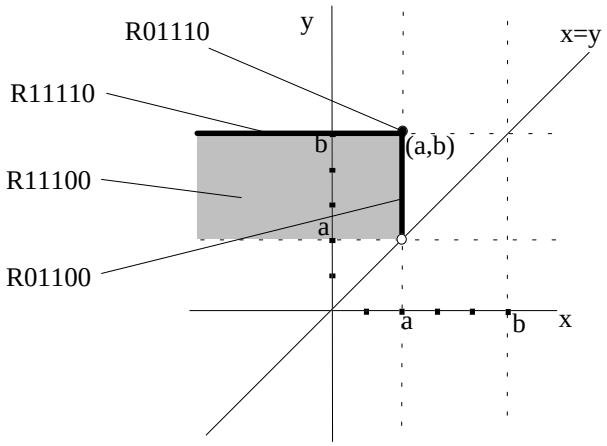

Figure 5 Geometric Representation of Interval Relations (assuming the reference interval [a,b])

Theorem 1. *Assume a reference interval* [*a*,*b*], *a primary interval* [*x*,*y*] *and a resolution scheme* $Y(a,b)=<Y_1,Y_2,\ldots, Y_m>$, *where* $seq(Y(a,b))= x_1x_2\ldots x_n$. *A relation R between* [*x*,*y*] *and* [*a*,*b*] *is convex iff its geometric representation can be defined by the conjunctive expression*

$$(a_1\ \sigma_1\ x\ \sigma_2\ a_2) \wedge (a_3\ \sigma_3\ y\ \sigma_4\ a_4) \wedge (x<y),$$

where σ_i *is either* $<$ *or* $\leq$, $a_i \in \mathbf{R}$ *and* $a_1<a_3$, $a_2<a_4$, $a_1\leq a_2$, $a_3\leq a_4$.

Proof sketch. The key idea is that, according to the above expression, *x* and *y* are constrained by convex interval domains, say *dom*(*x*) and *dom*(*y*). If *R* is convex then it is a relation interval $[R_1, R_2]$ at the corresponding lattice. Let Y_i,Y_j, be the leftmost and rightmost regions of interest whose corresponding bit is set to "1" in R_1 and Y_k,Y_l, be the leftmost and rightmost regions of interest whose corresponding bit is set to "1" in R_2. Since *R* can be any relation between R_1 and R_2, it can be expressed

by the conjunction:

$$(Y_i.l\ \sigma_1\ x\ \sigma_2\ Y_k.r)\wedge(Y_j.l\ \sigma_3\ y\ \sigma_4\ Y_l.r)\wedge(x < y)$$

For the inverse, assume that x and y are constrained by $dom(x)$ and $dom(y)$, given respectively by the expressions $(a_1\ \sigma_1\ x\ \sigma_2\ a_2)$, $(a_3\ \sigma_3\ y\ \sigma_4\ a_4)$. In a similar manner we can find appropriate regions of interest Y_i, Y_j, Y_k, Y_l, whose intersection with $dom(x)$ and $dom(y)$ is non-empty and construct two relation strings R_1, R_2 such that $R=[R_1, R_2]$ is convex (see [DH99] for the full proof). □

Assuming that we have stored in a database only points that correspond to valid 1D intervals (points above the $x=y$ line), the above theorem essentially says that a 1D selection query, asking for an approximate (i.e. convex) relation with respect to an interval $[a,b]$, is equivalent to a classical 2D range query. For the example in Figure 5 the approximate relation is $[R_{11100}, R_{01110}]$ (adopting the scheme of Allen's relations) and the corresponding query window is $(-\infty,a]\times(a,b]$.

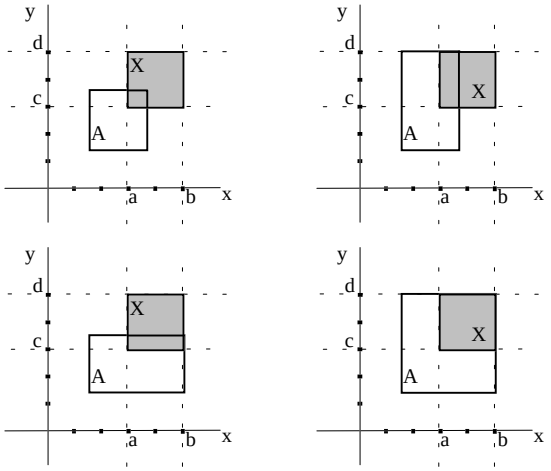

Figure 6 A 2D Approximate Query

For a 2D example, consider the four distinct queries of Figure 6 which are all instances of the approximate query relation $[R_{11100\text{-}11100}, R_{11110\text{-}11110}]$. This query is equivalent to the range query $(-\infty,a)\times(a,b]\times(-\infty,c)\times(c,d]$ in a 4D space where the rectangle X is represented by the 4-tuple <a, b, c, d>.

Generalising, an ND convex relation selection query is equivalent to a classic 2ND range query.

4 Data Structures For Range Queries

In this section we describe fast main memory and secondary memory structures for range queries. As explained in the previous section, selection queries can in general be mapped to ND (where N is even) range queries, where ranges can be the combination of N finite or semi-infinite, closed or open 1D intervals. The proposed structures accommodate all type for ranges, with appropriate adjustment of the searching algorithms.

4.1 Structures For Main Memory

For three-sided range queries on the plane (i.e. queries of the form $[a,b]\times(-\infty,c]$ - any of the four endpoints could be infinity) we can use the classic priority search tree [M84] that answers queries in $O(logn+k)$ time (n being the number of points and k being the output size), is updated in $O(logn)$ time and uses $O(n)$ space. On the grid (where objects are described by integer coordinates), Gabow et al. [GBT84] solved the static version of the problem in optimal query time $O(k)$ and $O(n+N)$ space (N being the universe size).

Their structure retains the $O(k)$ query time even for continuous space problems, provided we have access to the nodes of the structure corresponding to a, b (the endpoints of the closed range). In the sequel we will show how this structure permits to solve efficiently the 2D range searching problem.

Theorem 2. *In the RAM model of computation a set S of n points can be stored in an O(nlogn) space data structure that answers two-dimensional range queries in $O(\sqrt{logn}+k)$ time.*

Proof. Let $[a_1,b_1]\times[a_2,b_2]$ be the query range. We use a balanced binary tree T, which stores in its leaves the points according to their y-coordinate. To each internal node u we associate two sets of points S_1, S_2. S_1 contains the points of the left subtree of u while S_2 contains the points of the right subtree. Each of the S_1 is stored in a secondary structure $D_1(u)$ while each set S_2 is stored in a secondary structure $D_2(u)$. We use $D_1(u)$ to answer queries of the form $[a_1,b_1]\times[a_2,+\infty]$ and $D_2(u)$ to answer queries of the form $[a_1,b_1]\times[-\infty,b_2]$. $D_1(u)$ and $D_2(u)$ are implemented using the structure of [GBT84]. Moreover, the sets S_i are stored in two instances of Anderson's [A96] structure that permits 1D searching in $O(\sqrt{logn})$ time and linear space. Since each point is stored in $O(logn)$ secondary structures the space is $O(nlogn)$.

To answer a window query of the form $[a_1,b_1]\times[a_2, b_2]$ we find in T the node u where the search paths for a_2 and b_2 split. In other words, we find the nearest common ancestor of a_2 and b_2 in constant time [HT84]. Then we locate in S_1 and S_2 (using the [A96] structures) the a_1, b_1 values and query the structure $D_1(u)$ with $[a_1,b_1]\times[a_2,+\infty]$ and $D_2(u)$ with $[a_1,b_1]\times[-\infty,b_2]$. This takes time $O(\sqrt{logn}+k)$. □

For higher dimensions, the d-dimensional window query can be solved using the classic Range tree [M84] in $O(log^{d-1}nloglogn+k)$ query time, $O(log^{d-1}nloglogn)$ update time and $O(nlog^{d-1}n)$ space.

In static d-dimensional searching there are several related structures that can achieve less time and space. In the following, we describe in detail such a structure

because it exhibits a number of techniques common in the design of both main and secondary memory structures.

Let S be a set of n d-dimensional points, $d \geq 3$. We want to store them in a data structure so that the points that lie in a query range of the form $[a_1,b_1]\times[a_2,b_2]\times\ldots[a_d,b_d]$ can be found efficiently. We show that this problem can be solved in $O(log^{d-2}n+k)$ time and $O(nlog^{d-1}n)$ space. First we show how to solve the 3D problem for queries of the form $[a_1,b_1]\times[a_2,b_2]\times(-\infty,b_3]$. We construct a two-level structure. The first level is a balanced binary tree T that stores in its leaves the points of S sorted according to their first coordinate. With each internal node u we associate a secondary structure $D(u)$ for the points stored in the subtree of u. Each $D(u)$ is implemented as the structure of [GBT84] for the last two coordinates of the points.

Now consider a query. We locate the $O(logn)$ nodes in T that store the points with first coordinate in $[a_1,b_1]$. For each node u we query the structure $D(u)$ with the range $[a_2,b_2]\times(-\infty,b_3]$. Through fractional cascading [CG86], the leaves on the search paths for the values a_2, b_2 can be found in $O(logn+logn)=O(logn)$ time. So the query time is $O(logn+k)$. The space is $O(nlogn)$ since each point is stored in $O(logn)$ secondary structures and each secondary structure needs linear space. Therefore we have an overall query performance $O(logn+k)$, using $O(nlogn)$ space.

Now consider the general case of arbitrary ranges $[a_1,b_1]\times[a_2,b_2]\times[a_3,b_3]$. As in [O88] we use a technique of Edelsbrunner [E81]. Let T be a balanced binary tree which stores in its leaves the points according to their third coordinate. To each internal node u we associate two sets of points S_1, S_2. The set S_1 contains the points in the left subtree of u while the set S_2 contains the points in the right subtree of u. Each set S_1 is stored in a secondary structure D_1 while each set S_2 is stored in a secondary structure D_2. The D_1 structures answer queries with ranges of the form $[a_1,b_1]\times[a_2,b_2]\times(a_3,+\infty]$ while the D_2 structures answer queries of the form $[a_1,b_1]\times[a_2,b_2]\times(-\infty,b_3]$. Since each point is stored in $O(logn)$ secondary structures the space is $(nlog^2n)$. To answer a query with the range $[a_1,b_1]\times[a_2,b_2]\times(a_3,b_3]$ we find in T the node u where the search paths for a_3 and b_3 split. Then we query the structure $D_1(u)$ with $[a_1,b_1]\times[a_2,b_2]\times(a_3,+\infty]$ and $D_2(u)$ with $[a_1,b_1]\times[a_2,b_2]\times(-\infty,b_3]$. This takes $O(logn+k)$ time.

For the correctness note that since the search paths for a_3 and b_3 split in u, it follows that the points in $S_1(u)$ have third coordinate smaller than b_3 and the points in $S_2(u)$ have third coordinate greater than a_3. Furthermore all points with third coordinate in $[a_3,b_3]$ are in $S_1(u)\cup S_2(u)$ so the query algorithm is correct. Therefore:

Lemma 1. *The three dimensional static range searching problem can be solved in $O(nlog^2n)$ space and $O(logn+k)$ query time.* □

To solve the d-dimensional problem for $d>3$ we use a structure with $d-2$ levels. The first level is a balanced binary tree, which stores in its leaves the points according to their first coordinate. To each node we associate the set of points stored in its subtree. This set is stored, in the same way, in a second level structure according to the second coordinates of the points. The construction continues in the same way, until the last $(d-2)$th level, where we use the structure of Lemma 1. It follows easily that each level incurs an increase by a $logn$ factor to the query time and space of the structure, that is:

Theorem 3. *In the RAM model, the d-dimensional static range searching problem with iso-oriented rectangular ranges can be solved in $O(nlog^{d-1}n)$ space and $O(log^{d-2}n+k)$ query time.* □

Notes: i) The same result can be easily obtained in the Pointer Machine model [BKMT97]. ii) For RAM, Willard [W92] presented a structure that requires less space $O(nlog^{d-1}n/loglogn)$ but a little more time $O(log^{d-1}n/loglogn+k)$. iii) In the special case of 3D dominance searching (all ranges semi-infinite) a set of three linear space algorithms where proposed in [MT98]. The first works on a grid, is rather complicated and achieves $O((loglogU)^2logloglogU+kloglogU)$ time performance, where U the universe size of the set S of n points. The second achieves $O(lognloglogn+k)$ time performance for the pointer machine model of computation and the third achieves $O(logn+k)$ time performance for the RAM model of computation.

4.2 Structures For Secondary Memory

Let n denote the number of stored items and t the output size. We assume that our external memory consists of a single disk and that each I/O operation is able to transfer a block of B units of data from the disk to the main memory. The efficiency of an algorithm will be measured in terms of the number of the performed I/O accesses and the number of blocks required to store the proposed structure.

Although B-trees [C79] and their variations achieve optimal worst-case bounds for the 1D range searching problem, this is not the case for higher dimensions due to the inefficiency in mapping the successful main memory data structures to secondary memory. Moreover, the traditional spatial multidimensional indexing data structures (R-trees and variants, grid files,

quad trees, hB-trees, kd-B-tres, space filling curves, etc.) have good average case performance (validated usually through experimental tests), but their worst case behavior can be very bad.

Recently, a significant research effort has focused on obtaining external multidimensional range searching structures with satisfying worst case behavior. This research, triggered by the influential paper by Kanellakis et al. [KRVV96] aims in designing data structures with performance similar to the performance of the main memory data structures. Similar in this context means that the resultant structures should have query time of the form $O(log_B n+t/B)$ and the space usage should be close to linear ($O(n/B)$).

For the special case of 2D range searching with both ranges semi-infinite and with the query point lying on the diagonal of the *xy*-plane, Arge and Vitter [AV96] designed an optimal dynamic data structure which uses $O(n/B)$ space and supports queries and updates in $O(log_B n+t/B)$ and $O(log_B n)$ time, respectively. The proposed structure is an external version of the well known interval tree data structure [M84] and is based on the development of an elegant weight-balanced B-tree. The techniques developed there are of independent interest because they can be used to derive worst-case versions of other external and main memory data structures.

In [RS94] a technique termed *path caching* was developed that yielded a data structure that answers three-sided range queries in optimal $O(log_B n+t/B)$ query time, supports updates in $O(log_B n)$ amortized time and uses $O((n/B)loglogB)$ space. The technique is based on the movement of data from the internal nodes of a tree-like data structure to the leaves in order to facilitate the retrieval of information in a blocked fashion. The pathing technique was extended in [SR95] and a new dynamic structure called *p-range tree* was developed. The structure answers three-sided range queries in $O(log_B n+t/B+IL^*(B))$ time (IL^* denotes the iterated log^* function) supports updates in $O(log_B n+(log_B n)^2/B)$ amortized time and uses linear ($O(n/B)$) space. The structure can be modified to answer 2D range queries in the same query time bound and with optimal space $O(vlogv/loglog_B(n+1))$ ($v=n/B$).

Finally in [ASV99] a data structure was developed for the three-sided range query with optimal time, space and update bounds. The structure, which was based on the use of persistence and weight-balanced B-trees can be generalized to handle 2D range queries in $O(log_B n+t/B)$ query time, $O(vlogv/log(log_B n+1))$ space and $O((log_B n)logv/log(log_B n+1))$ update time ($v=n/B$).

For higher dimensions the related research is scarce. In [VV96] a near optimal static data structure for the 3D range searching was proposed. The structure relies on a geometric partitioning of the stored points (related to the well known concept of maximal layers) and with some modifications [V98] answers queries in optimal $O(log_B n+t/B)$ time and uses $O(v(logv)^k/(log(log_B n+1))^k)$ space. Here k denotes the number of closed ranges in the query and $v=n/B$.

Lately, Grossi and Italiano [GI98] designed a multidimensional version of B-trees called *cross-trees* that use linear space, have update time $O(log_B n)$ and answer queries in $O(n^{1-1/d}+t/B)$ time.

5 CONCLUSIONS

Spatiotemporal information is increasingly becoming an integral part of databases and knowledge bases. The need for its effective and efficient manipulation has motivated a lot of research efforts in the DB and AI areas. These have yielded several formal frameworks for the representation of binary spatiotemporal relations, which can be exploited by query processors for the efficient handling of spatiotemporal retrieval.

In this work we have focused on mechanisms for the efficient processing of spatiotemporal selection queries. Initially we formulated an innovative relation model which allows the description of spatiotemporal relations at varying detail (resolution) levels. The model is based on a binary string encoding of 1D relations and its straight forward extension to multiple dimensions.

The relations which are characterised at every resolution level have an inherent poset structure, called conceptual neighbourhood, which we exploit for the definition of approximate relations. An approximate relation, called a convex relation, is an interval in the lattice of such a poset structure and captures a continuous uncertainty in the representation of a particular relationship. Approximate relations constitute effective means to deal with approximate spatiotemporal selection queries. In particular, we have mapped every approximate ND selection query (with exact queries being a certain subset thereof) to an equivalent 2ND classic range query.

On these grounds we have proposed several existing and two new main memory and secondary memory structures that answer range queries in either optimal or the best known asymptotic time. The table in the next page summarises the performance behaviour of the proposed structures. The shaded fields indicate our new result for the static 2D range query problem and an extension of the techniques presented in [GBT84] for static range queries of higher dimensions. The rest

consist of the state-of-the-art structures for such problems. Unless explicitly stated otherwise, the time and space complexities refer to problems for continuous space.

Overall, the techniques proposed in this work are significant for GIS, temporal databases, spatial/ multimedia databases and other disciplines for which the handling and efficient querying of multidimensional data is of primary importance.

Some of the proposed structures are currently being implemented and the results so far are promising. Our next step for future continuation of this work is an extensive experimental evaluation of various structures for several classes of selection queries.

Range Query

	Main Memory			
	Static		***Dynamic***	
	Space	Time	Space	Time
d=2	$O(n\log n)$	$O(\sqrt{\log n}+k)$	$O(n\log^{d-1} n)$	$O(\log^{d-1} n \log\log n+k)$
d=3	$O(n)$	• $O((\log\log U)^2 \log\log\log U+k \log\log U)$ (dominance in a grid) • $O(\log n+k)$ (dominance in continuous space)		
$d\geq 3$	$O(n\log^{d-1} n)$	$O(\log^{d-2} n+k)$		
	$O(n\log^{d-1} n/\log\log n)$	$O(\log^{d-1} n/\log\log n+k)$		
	Secondary Memory			
	Static		***Dynamic***	
	Space	Time	Space	Time
d=2			$O(n/B)$ (3-sided range)	$O(\log_B n+t/B)$
			$O(v\log v/\log(\log_B n+1))$	$O(\log_B n+t/B)$
d=3	$O(v(\log v)^k/(\log(\log_B n+1))^k)$	$O(\log_B n+t/B)$		

REFERENCES

[A96] Anderson, A., "Faster deterministic sorting and searching in linear space" *37th Annual IEEE Symposium on Foundations of Computer Science*, 1996.

[All83] Allen, J., "Maintaining Knowledge About Temporal Intervals", *CACM*, 26(11), 1983.

[ASV99] Arge, L., Samoladas, V., Vitter, J.S., "Two dimensional indexability and optimal range search indexing", n *Proceedings of the ACM Symposium on Principles of Database Systems*, 1999.

[AV96] Arge, L., Vitter, J.S., "Optimal dynamic interval management in external memory", *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1996.

[BKMT97] Bozanis, P., Kitsios, N., Makris, C., and Tsakalidis, A., "New Results on Intersection Query Problems", *The Computer Journal*, Volume 40 Issue 1 (1997).

[BKSS90] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B, "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles", *ACM SIGMOD*, 1990.

[C79] Comer, D., "The ubiquitous B-tree", *ACM Computing Surveys*, 11(2) 1979.

[CG86] Chazelle, B., Guibas, L., "Fractional cascading I, A data structuring technique; II, *Applications*, *Algorithmica* 1, 1986.

[Chv83] Chvatal, V., *"Linear Programming"*, W. H. Freeman, 1983.

[DH99] Delis, V., Hadzilacos, Th., "Binary String Relations: A Foundation for Spatiotemporal Knowledge Representation", *Technical Report TR99-02-01*, Computer Technology Institute, Patras, Greece, 1999 (available at http://www.cti.gr/RD3/People/delis/tr990201.pdf.

[DMS99] Delis, V., Makris, Ch., Sioutas, S., "Efficient Spatiotemporal Retrieval", *Technical Report TR99.05.02*, Computer Technology Institute, Patras, Greece.

[E81] Edelsbrunner, H., "A note on dynamic range searching", *Bull. EATCS* 13, 1981.

[Ege97] Egenhofer, M.J., "Spatial Relations: Models and Inferences", *Tutorial 2, International Symposium on Large Spatial databases* (*SSD'97*), 1997.

[EH91] Egenhofer, M.J., Herring, J., "Categorizing Binary Topological Relationships Between Regions,

Lines and Points in Geographic Databases", *Technical Report*, Department of Surveying Engineering, University of Maine, Orono, ME, 1991.

[Fre92] Freksa, C., "Temporal Reasoning based on Semi Intervals", *Artificial Intelligence*, 54, 1992.

[GBT84] Gabow, H.N., Bentley J.L., and Tarjan, R., E., "Scaling and related techniques for geometry problems" *l6th Annual ACM Symp. on the Theory of Computing*, 1984.

[Gae95] Gaede, V., "Optimal Redundancy in Spatial Database Systems", *International Symposium on Spatial Databases* (*SSD*), 1995.

[GG98] Gaede, V., Gunther, O., "Multidimensional Access Methods", *ACM Computing Surveys*, 30(2), 1998.

[GI98] Grossi, R., Italiano, G., F., "Efficient cross-trees for external memory", In J.Abello and J.S. Vitter, editors, *External Algorithms and Visualization* 1998.

[Gut84] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching", *ACM SIGMOD*, 1984.

[HS95] Hoel, E., Samet, H., "Benchmarking Spatial Join Operations with Spatial Output", *VLDB Conference*, 1995.

[HT94] D. Harel, R. E. Tarjan, "Fast algorithms for finding nearest common ancestor", *SIAM J. Comp.* 13, 1994.

[Kau] Kautz, H. A., "Temporal Reasoning", *MIT Encyclopedia of Cognitive Science* (*forthcoming*), http://www.research.att.com/~kautz/papers-ftp/index.html.

[KRVV96] Kanellakis, P.C., Ramaswamy, S., Vengroff, D.E., Vitter, J.S., "Indexing for data models with constraints and classes", *Journal of Computer and System Sciences*, 52(3), 1996.

[Lig91] Ligozat, G., "On Generalised Interval Calculi", *American Association of Artificial Intelligence International Conference*, 1991.

[LM88] Ladkin, P., Maddux, R., "The Algebra of Binary Constraint Networks", *Kestrel Institute Technical Report KES.U.88.9*, 1988.

[M84] Mehlhorn., K., "Data Structures and Algorithms 3-Multidimensional Searching and Computational Geometry", Springer Verlag, 1984.

[MT98] Makris, C., Tsakalidis, A., "Algorithms for three-dimensional dominance searching in linear space" *Information Processing Letters*, 66(6), 1998.

[O88] Overmars, M. H., "Efficient data structures for range searching on a grid", *J. Algorithms* 9, 1988.

[PMD98] Papadias, D., Mamoulis, N., Delis, B. "Algorithms for Querying by Spatial Structure", *VLDB Conference*, 1998.

[PS94] Papadias, D., Sellis, T., "Qualitative Representation of Spatial Knowledge in Two-Dimensional Space", *VLDB Journal*, 3(4), 1994.

[PT97] Papadias, D., Theodoridis, Y., "Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures", *International Journal of Geographic Information Science*, Vol. 11(2), 1997.

[RS94] Ramaswamy, S., Subramanian, S., "Path Caching: a technique for optimal external searching" in *Proceedings of the 13th ACM Conference on Principles of Database Systems*, 1994

[SRF87] Sellis, T., Roussopoulos, N., Faloutsos, C., "The R^+-tree: A Dynamic Index for Multidimensional Objects", *VLDB*, 1987.

[SR95] Subramanian, S., Ramaswamy, S., "The P-range tree: a new data structure for range searching in secondary memeory", *ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[TPSS98] Theodoridis, Y., Papadias, D., Stefanakis, E., Sellis, T., "Direction Relations and Two-Dimensional Range Queries: Optimization Techniques", *Data & Knowledge Engineering*, vol. 27(3), 1998.

[TS96] Theodoridis, Y., Sellis, T., "A Model for The Prediction of R-Tree Performance", *ACM PODS*, 1996.

[TVS96] Theodoridis, Y., Vazirgiannis, M., Sellis, T., "Spatio-Temporal Indexing for Large Multimedia Applications", *3rd IEEE Conference on Multimedia Computing and Systems*, 1996.

[vBC90] van Beek, P., Cohen, R., "Exact and Approximate Reasoning about Temporal Relations", *Computational Intelligence*, 6, 1990.

[V98] Vitter, J.S., "External Memory Algorithms" invited paper in *European Symposium on Algorithms*, 1998.

[VV96] Vengroff, D.,E., Vitter, J.,S., "Efficient 3-d range searching in external memory", *ACM Symposium on Theory of Computation*, 1996.

[W92] Willard, D., "Applications of the Fusion Tree Method to Computational Geometry and Searching", *ACM-SIAM Symposium on Discrete Algorithms*, 1992.