MANAGEMENT OF LARGE MOVING OBJECTS DATA SETS: INDEXING, BENCHMARKING AND UNCERTAINTY IN MOVEMENT REPRESENTATION

Talel Abdessalem, E.N.S.T., Paris, Talel.Abdessalem@enst.fr
Cédric du Mouza, eq. VERTIGO, lab. CEDRIC, C.N.A.M. Paris, dumouza@cnam.fr
José Moreira, E.N.S.T., Paris, jmoreira@uportu.pt
Philippe Rigaux, L.R.I., Univ. Paris-Sud, Orsay, rigaux@lri.fr

Abstract
This chapter deals with several important issues pertaining to the management of moving objects datasets in databases. The design of representative *benchmarks* is closely related to the formal characterization of the properties (that is, distribution, speed, nature of movement) of these datasets; *uncertainty* is another important aspect that conditions the accuracy of the representation and therefore the confidence in query results; finally, efficient *index structures*, along with their compatibility with existing softwares, is a crucial requirement for spatio-temporal databases, as it is for any other kind of data.

INTRODUCTION

A lot of emerging applications (traffic control, mobile computing, vehicles tracking) rely on large datasets of dynamic objects. This proliferation is encouraged by mature technologies (for example, the Global Positioning System, or GPS) that provide online information on mobile devices and enable communication between a centralized system and mobile users. Most of the services that can be provided by the system to a user are based on the location of the latter at a given instant (for instance, the case of company searching for the taxi nearest to a customer calling on his mobile phone, or a tourist in his car looking for his next hotel). But, apart from these so-called location-based services that deal with the present or future (expected) positions of objects, we can also envisage applications that study the past movements within large moving objects datasets (for data-mining purposes, for instance).
These examples illustrate some new requirements that address the core functionalities of Database Management Systems (DBMS). Indeed, we must consider new data models (as any previously proposed model falls short in representing continuous movements), new query languages and new system-level support. In this chapter we focus on the latter aspect. More specifically, we propose a survey of the following issues: benchmarking of operations on large moving objects datasets, uncertainty in trajectories representation and database indexing. Let us be more specific by shortly developing each topic.

Benchmarking

In computing, a benchmark is the result of running a set of standard tests on one component or system to compare its performance and capacity to that of other components or systems. They are designed to simulate a particular type of workload, running actual real-world programs on the system “application benchmarks,” or using specially created programs that impose the workload on the component “synthetic benchmarks.” Application benchmarks are meant to be representative of real-world applications and potentially give a better measure of real-world performance. On the other hand, synthetic benchmarks offer a sizeable workload of data sets and operations, allowing testing individual components (such as indexing methods or hard disks) and stressing the strengths and weaknesses of each one individually. In the spatio-temporal database context, benchmarks help to experiment with new approaches (for example, new languages or new indexing structures); they can be used to assess the effectiveness of a new system; and finally, they contribute to characterizing the properties of datasets.

Uncertainty
The representation of objects’ movement is inherently imprecise (Pfoser & Jensen 1999; Trajcevski, Wolfson, Zhang & Chamberlain, 2002). Imprecision is introduced by the measurement process in the sampling of positions and by the sampling approach itself. The accuracy of measurements depends largely on the instruments and the techniques used (consider the example of the GPS). These devices are only able to capture the movement of an object by sampling its position at discrete time points and, consequently, the exact position of an object between measurements is unknown. This feature, commonly referred to as uncertainty, gives rise to several problems regarding the representation and querying of moving objects. We shall discuss in thischapter the factors that determine the imprecision, and then study (in a database perspective) ways to handle this imprecision.

Indexing
The existence of efficient access methods is one of the most important features of modern database systems. Given the huge amount of data stored in such systems, there is indeed a crucial need for structures that allow filtering of irrelevant data during query processing. B+-tree and hash-based techniques are used quite intensively in the traditional relational DBMS context. It is a well-known fact for database practitioners that a query execution plan, which relies on an index, is several orders of magnitude faster than a plan that merely scans the database. It turns out, however, that these structures fall short in supporting queries over spatial or spatio-temporal data. We shall examine in this chapter the difficult challenges raised by indexing objects whose location changes continuously, describe some representative attempts to solve the problem and discuss research perspectives in this area.

The main objectives of this chapter are to provide an up-to-date panorama of the ongoing research devoted to moving objects management systems, with a strong emphasis on the aspects that determine the efficiency and reliability of such systems. We will successively investigate benchmarking, uncertainty and indexing, giving for each topic some concrete examples, a discussion on the raised problems, some general design guidelines commonly adopted to solve the problem and finally, a presentation of future trends together with the most recent references.

BENCHMARKING

We begin with a short introduction on the general issue of database benchmarking, and then study some representative benchmarks for spatio-temporal databases.

Background

The two major components of a benchmark are workload specification and measurement specification.
In database or transaction processing environments, the workload specifies the data and query sets. The data sets are used to populate the database. They can be composed of real-world data or produced by a data sets generator according to specific statistical models. The query sets simulate the activity occurring in the database, such as operational and decision support transactions, or batch jobs. A transaction set driver may be used to simulate environments, where a number of users input and manage queries or transactions via a terminal or desktop computer connected to a database, with “thinking” and “keying” times interleaved.
A measurement environment must specify a metric and a reporter. By definition, a metric for a feature is an association of numeric values to feature values in such a way that the general properties of distances are verified. In a benchmarking environment, a metric is required to confer significance to the performance evaluation results. An example of a metric is the number of transactions per second (TPS). The reporter specifies how to collect all relevant traces and logs and computes indices pertinent to the specific metrics. It must provide the detailed information required to make accurate decisions on the performance capability of a system under test.

All testing processes require a well-designed execution plan. Execution plans ensure real-world environments duplication during a benchmark. The results should not depend on foreign factors(such as the hardware and software configurations) that are not related with the components in evaluation. These configurations would be barely reproducible in other environments; therefore, the results obtained would be hardly or not even comparable with the results of a similar test in different settings.

Another important feature of a benchmark is to provide a model that is representative of real-world applications with an extensible workload, made of sizeable data sets and sets of queries with varying complexities. This ensures that the model is useful and yet verifiable. Portability (it should be easy to implement on a broad range of DBMS) and simplicity (it must be understandable), also are important qualities of a benchmark.

Benchmarks for moving objects databases

Benchmarking spatio-temporal systems is a novel issue; so far, emphasis has been on the development of spatio-temporal data sets generators. There are now data sets generators for simulating objects moving freely, with no or few restrictions in the movement of the objects, and generators for simulating the movement of objects for which the movement is constrained by a defined network, such as a road network.

Non-network-based generators

The first spatio-temporal data sets generator for moving objects has been the so-called Generator of SpatioTemporal Data (GSTD) (Theodoridis, Silva & Nascimento, 1999a) and later, a new version was proposed introducing some important new features (Theodoridis et al., 1999b). In its current version, GSTD is a Web-based application and there also are data sets, in XML format, that can be downloaded from the Web.

The GSTD allows the generation of data about points and MBRs to be moving on a rectangular space. The space can be populated by static spatial objects that obstruct the movement of the objects. The objects, points or rectangles, are initially distributed in space according to Uniform, Gaussian or Skewed distributions. The evolution of spatial objects is directed through the definition of a set of parameters that control the duration of an object instance (which involves changes of timestamps between consecutive instances), the shift of an object (which involves changes of spatial location in terms of shift/translation of center point) and, when generating MBRs, the resizing of an object (changes in object size).

The combination of possible different distributions for these parameters allows simulating different scenarios, such as objects moving equally slow or fast and uniformly on the map, having a relatively large number of slow objects moving randomly, or having a set of objects that converge to some area of the workspace or moving to some direction (east, for example). The cardinality of the data sets is assumed to be constant during the generation process. The generated data sets are memory-less, that is, future events do not depend on past states. This framework also defines how to handle objects that fall outside the map. Three alternatives are proposed: the adjustment approach, where coordinates are adjusted to fit the workspace; the toroid approach, where the objects that traverse one edge of the workspace enter back in the opposite edge; and the radar approach, where coordinates remain unchanged, although fall beyond the workspace.

The main goal of previous approaches was to produce data sets that are rich from the statistical point of view, but a question arises of how to generate datasets representative of the behavior of real-world objects.

The Oporto framework (Saglio & Moreira, 2001) presents the specification of a realistic spatio-temporal datasets generator. The motivation for this proposal is that real-world entities do not have a chaotic behavior. They are guided by goals and they do not ignore the environment around them; that is, they are sensitive to agents favoring certain kinds of behavior and to agents inducing them to avoid other kinds of behavior. So, the generator exploits a scenario with harbors (static points), spots (regions with fixed center and changing shape, representing plankton areas or storm areas), fishing ships (moving points) and shoals of fish (fully moving regions). The fishing ships move in the direction of the most attractive shoals of fish while trying to avoid storm areas. The shoals of fish are attracted by plankton areas.

The default generation model parameters are based on information obtained from a real application for monitoring fishing activities. These parameters are organized in three classes: sizing parameters, responsible for the size of the data sets; distribution parameters, responsible for the variations in temporal and spatial distribution of moving points; and miscellaneous parameters, responsible for a few realistic features. It is proposed to use a logarithmic scale for sizing the data sets, simulating 1, 10 and 1000 weeks of fishing activities, and two different scenarios – inshore and open-sea fishing – for the spatio-temporal distribution of data sets.

Network-based generators
Previous approaches do not consider applications where moving objects follow a given network. This issue has been covered by Brinkhoff (2000, 2002). This generator combines real data (the network) with user-defined properties for controlling the functionality of selected object classes. Important aspects are the maximum speed of connections, the influence of other moving objects or external impacts (for example, weather conditions) on the speed, the adequate determination of origination and destination of an object and time-scheduled traffic.

The generation of datasets requires three steps: the preparation of the network, which eventually involves the conversion of files describing the nodes and the edges of the network into the file formats supported by the generator; the definition of functions and parameters according to the environment of the system under test; and the visualization of the generated datasets (they are stored in a text file) using a tool that allows visualizing the motion of the objects.

As it is argued that it would be difficult to establish an environment where all these aspects could be defined by simple user interaction or by predefined parameters, the framework only supports a few standard parameters, and the specification of elaborate behavior for moving objects requires user-defined functions and parameter settings. The implementation of the generator is based on the Java programming language. The classes are predefined; only their functionality must be adapted.

Future trends

Research on benchmarking moving objects databases systems is a recent issue and, so far, the focus has been on the generation of synthetic data sets. There are now several applications available on the Web that allow generating free (Theodoridis et al., 1999b; Saglio et al., 2001) and network-based (Brinkhoff, 2002) movements, according to a diversity of rules and control parameters. The generated movement data sets are basically sequences of temporally ordered observations, each one representing the location of a moving object at a certain instant. These data sets can be used to populate a database storing the past movement of objects, or to simulate transactions for updating the last-known location on systems concerned with present and near-future positions of moving objects.

Works on the specification of query sets is quite limited and deserves attention in the future. Apart from the benchmark database queries for location-based services (Theodoridis, 2003), there is no other systematic approach in this area. The metric that has been used in the experiments published so far was the number of disk blocks read for the evaluation of some operation. Notice that, as moving objects database systems are not commercially available yet, the experiments performed have focused exclusively on evaluation of the performance of specific access methods and algorithms for spatio-temporal operations, usually a spatio-temporal windowing or clipping. Authors use their own data and query sets and execution plans; hence, it is very difficult or even impossible to compare the performance of the different methods and techniques that have been proposed. This important issue should be considered in the near future by researchers in this area.

UNCERTAINTY

Let us now turn our attention to the uncertainty of moving objects trajectories. As mentioned in the introduction of this chapter, the history of the objects' movement is inherently imprecise (Pfoser et al., 1999; Trajcevski et al., 2002). Imprecision is introduced by the measurement process in the sampling of positions and by the sampling approach itself. We begin with a short introductory part that illustrates the issue with an example and discusses the factors that determine the imprecision. We then study, in a database perspective, how to handle this imprecision.

`The uncertainty of moving objects trajectories` Consider the concrete case of a port authority dealing with a spread of toxic waste in the sea and querying a nautical surveillance system to know which ships have crossed the polluted zone for a specified time interval. Imagine that the ship responsible for the waste has actually followed the trajectory represented in Figure 1. The dots represent observations made during the specified time period, the shaded region represents the polluted area and the hatched line a trajectory that might have been inferred from the observations.

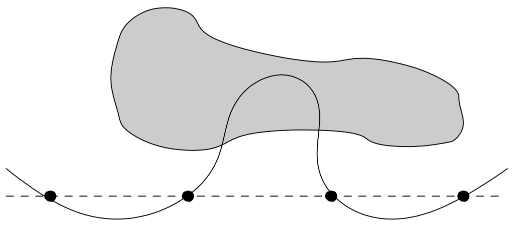
Figure 1. Indeterminacy of the behavior of an object between consecutive observations

The hatched line does not cross the shaded region and, thus, an answer to the query based on this estimation of the trajectory would not include the guilty ship. On the contrary, an answer may include false candidates whose inferred trajectory crosses the area even though they have not actually been there.

Uncertainty of past, present and future positions
The preceding example focuses on the history of objects' movement. In general, the focus may be put on the past movement or on the future movement of objects, depending on the considered application. Different needs were identified, giving rise to two main approaches.

The first approach (Pfoser et al., 1999), focusing on past movements, addresses the needs of mining applications of spatio-temporal data: traffic mining, environment monitoring, and so forth. In this case, uncertainty is determined using the observed successive positions of objects and some known constraints on their velocity.

In the second approach (Sistla, Wolfson, Chamberlain & Dao, 1998; Wolfson, Sistla, Xu, Zhou, Chamberlain, Yesha & Rishe 1999c; Trajcevski et al., 2002), the focus is put on the uncertainty about the future movement of objects. This approach addresses the needs of real-time applications and location-based services. Uncertainty, fixed in advance, here is used to avoid frequent updates to the database when the actual object's trajectory deviates from its representation in the database. The database is not updated as long as the object's movement deviation is less than the permitted uncertainty.

Bounding uncertainty
There are physical constraints on the movement of objects, allowing limiting uncertainty of their position. Particularly, the uncertainty interval for an object moving on a road is a section of the road, whereas it is an area in the considered space for an object moving freely. When it comes to future movements in a two-dimensional space, the uncertainty area is a circle centered on the expected location of the object. Each circle bounds for a given instant the permitted deviation of an object. Objects are committed to send a location update when the deviation reaches the bound.

For past movements, since the positions between two consecutive samples are not measured, the best possibility is to limit the possibilities of where the moving object could have been (Pfoser & Tryfona, 2001; Pfoser et al., 1999; Moreira, Saglio & Ribeiro, 1999).

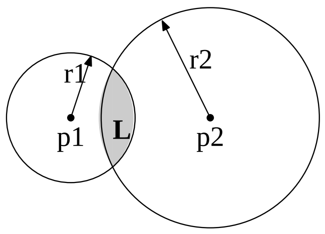

Figure 2. An lens area for a time instant

Let us consider a moving object *m*, a time instant *t* in an interval (t_1,t_2) between two consecutive observations (p_1,t_1) and (p_2,t_2) (ee Figure 2). We denote by p_1 *and* the positions of the moving object at the observation time instants t_1 and t_2, respectively. At time *t*, the distance between *m* and p_1 is inferior to $r_1 = vv_{max} \times (t- t_1)$ where vv_{max} is a user-defined value standing for the maximum velocity of moving object *m*. Distance between between *m* and p_2 is inferior to $r_2 = vv_{max} \times (t_2- t)$. So, at *t*, the moving object might be at any location within the area defined by the intersection of the two circles of radius r_1 and r_2. This is a so-called lens area (Pfoser et al., 1999) representing the set of all possible locations for a moving object at a certain time instant.

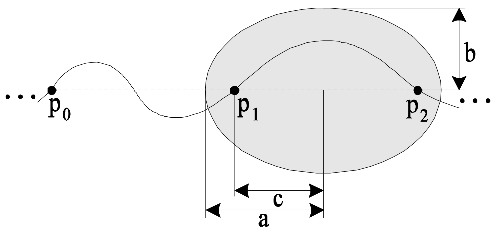

Figure 3. Coverage of all possible trajectories between two consecutive observations

The set of all locations where a moving object might be between two consecutive observations corresponds to an ellipsis (Figure 3). This ellipsis covers all the possible lens areas between the two consecutive observations. Its parameters *a* and *b* may be computed as follows: $a= vv_{max} \times (t_2- t_1)/2$, $c=(p_2 -p_1)/2$ and $b^2=a^2-c^2$.

`Dealing with uncertainty in ST databases` Data collected by sensor systems allow estimating the location of observed objects at any time instant between observations, assuming, for example, that the movement is linear and uniform between two consecutive observations. This semantics is not satisfactory to answer queries where uncertainty is significant. Hence, considering the uncertainty areas as described in the previous section, we propose to combine basic operations on moving objects with different semantics to provide meaningful operations in this context (Moreira et al., 2000).

Possibly Semantics

When considering the Possibly semantics, we look for the set of all possible candidates matching a query. This may be indicated to the query evaluator by adding a prefix "Possibly" to the chosen operation. Answers to such queries are supersets of the ideal results obtainable in an infinite precision representation. The answer to a query such as "Which are the objects that have been in Area C" includes all the objects that actually have been in that area, and it may also include some objects that have not. Similarly, is the answer to a query like "Give me the movement of object O

within Area C" includes all the parts of the movement of object O for which it actually was within Area C, and it may also include some parts of the movement for which the object was not there.
The set complement of this result consists in the values that definitely do not match the query predicate.

Surely Semantics
When considering the Surely semantics, only values that surely match the query are returned. In that case, the prefix "Surely" is associated to query operations, and the answers represent subsets of the reality. The answer to a query such as "Which are the objects that have been in Area C" would be a subset of the objects that actually have been in Area C. This means that some objects that actually have been in the area as well may be ommited by the answer. Similarly, the answer to a query like "Give me the movement of object O within Area C" would include only some parts of the movement of object O for which it was actually within Area C. Some parts for which object O also was within Area C may be ommited.

The set difference of the result of an operation using the Possibly semantics with the result of the same basic operation using the Surely semantics returns a set of values for which it is neither possible to assert that they do match the query predicate nor that they do not.

Using probabilistic methods
Consider now the use of methods that allow estimating the location for a moving object at any time instant, specifically, a probabilistic distribution method over the uncertainty areas. Query expressions evaluation can be augmented with a probabilistic estimation of the answer. The goal is to be able to answer queries such as, "Which are the objects that have a probability of 0.6 to be inside Area C," or "Which were the ships in a certain area during a given time interval, with a probably of at least 30%."

Figure 4 illustrates this query (Pfoser et al., 1999). We consider the case of a future movement in a two-dimensional space. As described in the uncertainty of moving objects trajectories section, the lens area bounds the permitted deviation of the object. If we suppose the probability distribution to be uniform in this lens area, then the object is said within a given area (query window) with a probability of 30%, if at least 30% of its lens area is concentrated within that area.

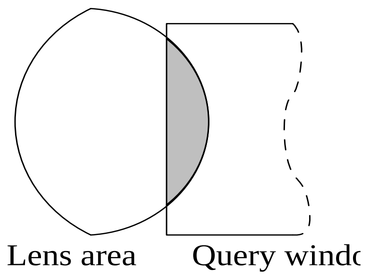

Figure 4. Probability of intersection between a lens area and a query window

In the following query examples, prefix "Proba" is used to return the measured probability for the evaluated predicate. When a probabilistic method for the measurement of uncertainty is used, the Possibly and Surely semantics may be seen as particular cases.

Query examples

We present here some query examples to illustrate the semantic variants proposed above. A more detailed description of relevant query operations may be found in Moreira et al. (2000).

We consider as a case study the MONICAP system for monitoring and control of fishing activities (CCMP, Inesc). The system has been used since 1992 by the Portuguese general authority for fishing activities (IGP). It continuously monitors the position of the vessels and records the history of their courses. Vessels are represented as moving objects. Static objects represent fishing areas, harbors, and so forth.

Consider the following relations:
FishingShips(reference:string, name:string, voyages:movement)
ForbiddenAreas(name:string, geometry:polygon)

Notice that the entire movement of a vessel is represented as an attribute in the relation *FishingShips*. The queries correspond to the kinds of questions that IGP would like to be able to answer based on their database system. They will be expressed here using an SQL-like syntax.

Q$_1$: Suppose the authorities want to investigate who was responsible for a spread of waste in the sea and want to know the behavior of all vessels that could have been in the area.

```
SELECT x.name, PossiblyIn(x.voyages, :PollutedArea)
FROM FishingShips x
WHERE notEmpty(PossiblyIn(x.voyages, :PollutedArea));
```

The value *PollutedArea* is a user-defined polygon representing the area where the spread of waste has occurred. The expression *x.voyages* represents the movement of each fishing ship *x*. Operation *PossiblyIn* applied to the couple *(x.voyages, :PollutedArea)* returns, for each ship *x*, the part of its movement that *possibly* occurs inside the given area (that is, where lens areas intersect the *PollutedArea)*. The predicate *notEmpty()* determines whether this argument is empty. Here it allows selecting only the fishing ships that have a nonempty movement inside the *PollutedArea*. Query **Q$_1$** returns pairs of values, where the first is a name of a ship, and the second is the part of its movement that possibly occurred in the considered area.

Q$_2$: Authorities apply penalties when they are able to guarantee that a fishing ship has been in a forbidden area. The following query returns the name of the ships that have been in the Blue Coast reservation.

```
SELECT x.name
FROM FishingShips x, ForbiddenAreas y
WHERE y.name = “Blue Coast reservation”
AND notEmpty(SurelyIn(x.voyages, y.geometry));
```

Operation *SurelyIn* returns the parts of the movements for which it is possible to assure that a fishing ship was inside the forbidden area. Applying the predicate *notEmpty* to the result of the previous operation allows selecting the tuples of the required fishing ships.

In the following, we assume that a probabilistic method for measuring uncertainty is implemented on the MONICAP system. In that case, the previous two queries may be expressed as follows:

Q$_1$:

Q$_2$:

```
SELECT x.name, In(x.voyages, :PollutedArea)
FROM FishingShips x
WHERE ProbaIn(x.voyages, :PollutedArea)>0
```

```
SELECT x.name
FROM FishingShips x, ForbiddenAreas y
WHERE y.name = "Blue Coast reservation"
AND ProbaIn(x.voyages, y.geometry)=1
```

The *ProbaIn* operation, applied to its couple of arguments, returns a value between 0 and 1 corresponding to the measured probability that movement *x.voyages* may occur inside the given area.

Q_3: We search here for the fishing ships that were closer than 0.2 miles from vessel "P01" on May 28, 2000 with a probability greater than 0.6.

```
SELECT x.name
FROM FishingShips x, FishingShips y
WHERE x.reference = "P01"
AND y.reference ≠ x.reference
AND 0.6 < ProbaWithinDistance(During(x.voyages,⟨05/28/2000,05/29/2000⟩),
                              During(y.voyages,⟨05/28/2000,05/29/2000⟩),0.2));
```

The first condition allows selecting the fishing ship with reference "P01." The second avoids comparing the distance of "P01" with itself. Finally, the third condition restricts the selection to a measured probability greater than 0.6 from all fishing ships that possibly were at a distance inferior to 0.2 miles from "P01" during May 28, 2000.

Future trends

In recent years, uncertainty handling emerged as an important issue in moving object database research. Several aspects were investigated. Two complementary models were given: Pfoser et al. (1999) focused on past objects' movement when Wolfson et al. (1999c) and Trajcevski et al. (2002) treated future objects' movement. Wolfson et al. (1999b, 1999a) investigated the communication cost of uncertainty in the case of a real-time application. Pfoser et al. (2001) added fuzziness in object location and considered the case of moving objects that may change their geometry in time.
An important issue of the current research activity in this domain is the design of a probabilistic model of uncertainty. The goal is to handle more realistic (non-uniform) distributions of probability on the location of moving objects, and to be able to measure the validity of the query answers. Recent results (Cheng, Prabhakar & Kalashnikov, 2003; b, Cheng, Kalashnikov & Prabhakar, 2003a; Trajcevski et al., 2003) are going toward this goal, even if they just briefly touch upon the possibility of a non-uniform distribution.

INDEXING

In this section we investigate the shortcomings of traditional structures with respect to spatio-temporal databases indexing. We present some indexing techniques that have been recently proposed to overcome these limitations, and discuss the perspective of ongoing research.

General issues – Background

Since traditional structures cannot be used for multidimensional data indexing, during the last two decades there have been a lot of works to design efficient and reasonably simple spatial indices, like the R-tree, that can be used in existing DBMSs to support the optimization of spatial queries (see the surveys in Gaede and Guenther, (1998) and Rigaux, Scholl and Voisard, (2001)). R-trees rely on a balanced hierarchical structure in which each tree node, whether internal or leaf, is mapped onto a disk page. The R-tree (and its variants) organize rectangles (which constitute the bounding boxes of the objects in the indexed dataset) according to containment properties.

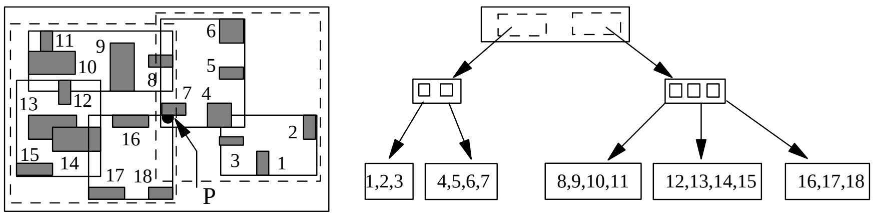

Figure 5. R-tree indexing

An example of an R-tree is shown in Figure 5. We assume there that a disk page can store no more than four entries (an entry is a pair composed of a point and a bounding box, and is used to navigate in the tree). Groups of four or less objects are then created and assigned to pages in the structure, based on their proximity relationships. This leads, on our example, to the groups *{1,2,3}, {4,5,6,7},* and so on. Note that grouping objects close in the space aims at minimizing the overlapping of the groups' bounding boxes and helps reduce the disk accesses during search operations. The same grouping process is applied to the bounding boxes of groups, recursively, until one obtains a single disk page, the *root* of the tree.
The R-tree properties are similar to that of the B-tree; that is, the tree is balanced, its size is logarithmic with the size of the indexed data set and its space complexity is linear. It supports point and window queries. A point query, for instance, is performed by a top-down traversal of the tree, exploring at each level the sub-trees whose bounding box contains the argument point. R-trees extend B-tree indexing to multi-dimensional data. It relies, however, on the important assumption that these data remain constant once stored in the database, until explicitly updated. In the presence of objects moving in the plane, this assumption is no longer valid, as it would require very continuous updates to the structure.
We chose to focus the rest of this section on a representative proposal, the Time-Parameterized R-tree (TPR-tree), and describe its design, its properties and the queries it supports. We use this first presentation as a basis for a more general discussion devoted to the challenges raised by moving objects indexingand to the issues that remain to be solved by current and future research.

A Detailed Example: The TPR-tree

The TPR-tree (Saltenis, 2000) is an extension of the R-tree that aims at indexing current and future positions (but not the past ones) of moving objects. More precisely, the index handles any object whose position is a tuple of coordinates $(x_1(t),\ldots,x_d(t))$. Each coordinate $xi(t)$ is itself a linear function of time of the form $xi(t)=xi(t_0)+v_i(t-t_0)$, where the instant t_0 defines the reference position of the indexed object and v_i is the speed of the object along the axis i.
Note that using linear function means that we consider only objects with constant speed, which is a reasonable assumption. In the following we restrict the discussion to objects moving in the 2D plane ($d=2$).

Building a TPR-tree

Given a dataset of objects whose trajectories comply with the above representation, the TPR-tree is an R-tree-like index, built at time t_0 and valid for a time interval U. The basic idea of the structure is to construct an R-tree with time-evolving bounding boxes. Similar to the classical R-tree, each leaf corresponds to a bounding box that contains a group of objects *{o_1,…,o_n}*. But unlike the R-tree, the edges of a bounding box in the TPR-tree “move” so as to enclose as accurately as possible its associated group of objects during all the lifetime of the index, *[t_0, t_0+U]*.

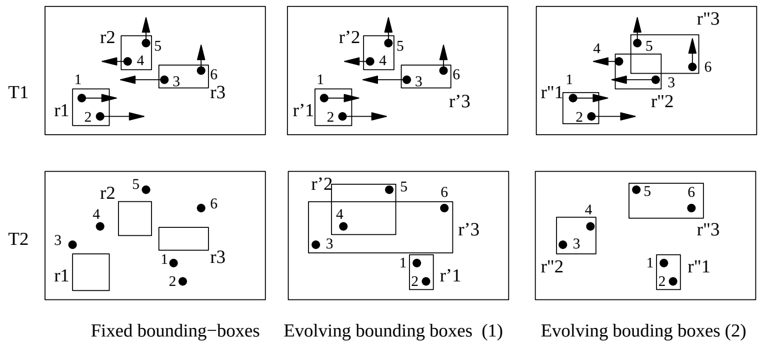

Figure 6. Evolving bounding boxes in the TPR-tree

Figure 6, inspired from Saltenis, Jensen, Leutenegger and Lopez, (2000), illustrates this intuition. We consider six moving objects and assume that the capacity of a disk page is two objects. The upper part of the figure shows the positions at time T_1, and the lower one, the positions at T_2 (T_2>T_1). An arrow is associated with each object, showing its direction and speed. Each column corresponds to a possible solution for indexing these data with bounding boxes. The left side shows the classical R-tree approach, with three fixed bounding boxes, *r1 ,r2 ,r3*, determined at time T_1. It appears clearly that this approach is not adapted since a bounding box at T_1 is obsolete at T_2 (for instance, r_1 no longer includes any object at T_2). This approach requires updating the bounding boxes so frequently that the maintenance of the index becomes impossible.

Consider now evolving bounding boxes, that is, rectangles whose edges move along the two axes according to a linear function of time. The choice for clustering objects in a box should now take into account not only their spatial proximity at time T_1 but also their future positions. For instance, the central column of Figure 6 shows that grouping objects by merely considering their closeness at time T_1 gives a good result for *r’1*, but not for *r’2* and *r’3*. The index greatly suffers from the increased overlapping.

The rightmost column illustrates a satisfactory grouping of our six objects, which takes in account their proximity along the whole validity interval of the index. This leads to grouping together objects that share more or less the same direction and the same speed. The comparison of the evolving boxes *r’’1, r’’2, r’’3* at T_1 and T_2 shows the superiority of this approach over the previous one.

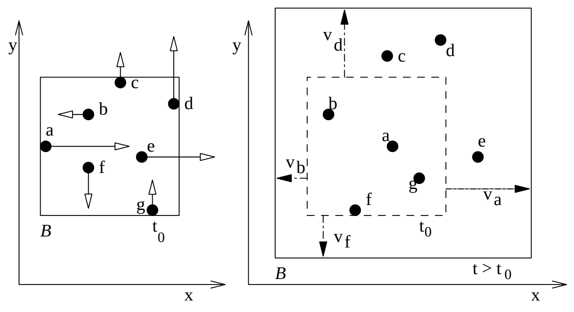

Figure 7. Managing the evolution of a bounding box

Figure 7 gives an example of the evolution of the bounding box between t_0 and some $t>t_0$. The position and speed of each object are known at t_0. To find the growing speed of B we determine the minimal and maximal speeds on both the x and y axes. For instancem the right border of the bounding box moves with a speed that corresponds to the maximal values of the projection on the x axis of all the objects inside the bounding box, here $v_a^x(t_0)$, and the speed of the left border corresponds to the minimal speed, here $v_b^x(t_0)$. The bounding box B of a TPR-tree is minimal at time t_0. Each edge of B moves along each axis x_i with a speed v_i, which is equal to the maximal speed along x_i of the objects contained in B. It follows that the minimality of the bounding box is not preserved.
In a classical R-tree, the insertion, update and deletion strategies aim at minimizing the bounding box area, the overlapping of the bounding boxes and the perimeters. The TPR-tree maintenance uses a similar approach, using the integral version of these parameters. Intuitively, the minimization always considers the (continuous) sum of the parameters' values over the validity interval of the tree. Refer to Saltenis et al. (2000) for details.

Queries supported by the TPR-tree

The index supports three kinds of queries:

- spatial window queries at a given instant \Delta
- spatial window queries for a time interval $[t_{beg}, t_{end}]$
- moving window queries, with values R_{beg} at t_{beg}, and R_{end} at t_{end}

Figure 8 presents examples for these three kinds of queries in a 1D-space. Consequently, a spatial window query is here at a given instant a segment (so are R_{beg} and R_{end}). In this example, o_x denotes a moving object and we assume that $U=1$, that is, a new index is built every time unit.

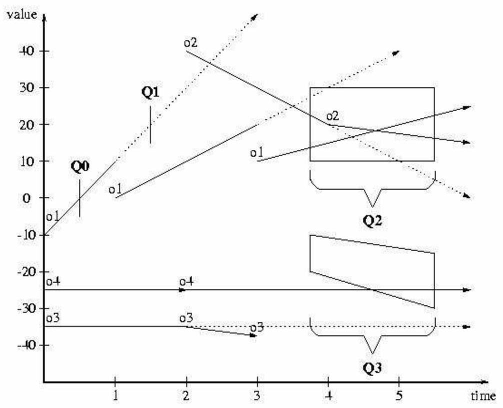

Figure 8. Examples of queries in a 1D-space

1) $Q_0=([-5,5],0.5)$ and $Q_1=([15,25],1.5)$ are queries of the first kind; however, two cases appear:
   a) either the query is formulated before t_0 (e.g. Q_0), that is, before the update, and the result is consequently $\{o_1\}$, whose future trajectory is forecasted;
   b) or the answer is the empty set since the new position and speed of o_1 are now considered (for example, Q_1).
2) $Q_2=([10,30],3.75,5.5)$ is a query of the second category. If the query is expressed at a time $t<1$, the result is the empty set, since the expected trajectory for o_1 does not go through the query window, and object o_2 is unknown. If Q_2 is expressed at time $1<t<2$, the answer is o_2 since after

the update the planed trajectory intersects the query window. Object o_2 is still ignored. Finally, if Q_2 is expressed at a time $t>2$, the query result is the set $\{o_1, o_2\}$.

3) Q_3 belongs to the third category and its result is the object o_4 at any time.

If we consider the query of the first category, the search with the TPR-tree is close to that of the R-tree: We select a bounding box *B* if the query window *R* overlaps *mbb*. For the queries of the second and third categories, defined on a time interval $[t_{beg},t_{end}]$, we have to select the bounding boxes that intersect the query window between t_{beg} and t_{end}. The algorithms rely on the observation that two moving rectangles intersect if there is an instant $t \in [t_{beg}, t_{end}]$ such that their projections on each dimension intersect.

In summary, the index is valid only for a period *U*, and its performances deterioriate with time because the bounding boxes grow. A new index must be created whenever the validity period is over. In other words, it is not a fully dynamic structure that can be created once and adapts to the evolutions of the database.

Discussion and future trends

Let us now study the characteristics of the TPR-tree and evaluate to what extent it meets the general needs of spatio-temporal indexing. Throughout the discussion we will mention other techniques and recent proposals that apply to other areas, or provide an alternative approach.

Indexing past, current and future positions

The TPR-tree indexes the current and the future positions of moving objects, and is therefore relevant for the applications that track, in real time, objects equipped with a positioning system. This structure is the state-of-the-art solution for this particular situation. Its performances have been analyzed recently in Tao, Papadias and Sun, (2003), which also describes important improvements to the construction algorithms. Among the other proposals that address the same problem, but are somewhat less satisfactory, we can mention Tayeb, Ulusoy and Wolfson, (1998), which uses PMR-quadtrees to index one-dimensional moving points –(thus, two indices are needed for 2D points), and several theoretical studies, such as Kollios, Gunopolos and Tsotras (1999) and Agarwal, Arg and Erickson (2000), which, unfortunately, do not provide a practical solution.

Another quite relevant area of research concerns the indexing of past locations, which can be of interest to data-mining applications (for example, an application that analyzes the movements of a given population in a given area and for a given period). If we keep the assumption that movements can be decomposed in a finite number of consecutive time intervals, and that for each interval the speed of an object is constant, then the problem is to index a polyline in a 3D space, with “time” as a third dimension.

A straightforward solution is to build a 3D R-tree, as proposed in Theodoridis et al. (1996). Note that this assumes that the bounding boxes are bounded; that is, that each time interval associated to the segments of the polyline are closed. Otherwise, one of the bounding boxes is of infinite size, and this raises problems. This can be compared with the TPR-tree, which considers only one segment, and bounds the time interval of interest $[t_0, t_0+U]$. Another possibility is to rely on a set of R-trees, each covering a time interval. The approach is first proposed in Nascimento et al.(1998; 1999), with a structure called the HR-tree that maintains an R-tree for each timestamp. The trees of the previous timestamps are never modified. In order to save space, the common branches of consecutive trees are stored only once. The HR-tree performs well for moving objects that frequently update their motion, but the performances are poor in range queries.

Several other proposals are worth mentioning – Tao et al. (2001); Pfoser et al. (2000); Porkaew, Lasaridis and Mehrotta, (2001); Hadjieleftheriou, Kollios, Tsotras and Gunopoulos (2002); and Saltenis et al. (2002), whose common approach is to extend R-trees to handle a polyline in a 3D space, with frequent updates that affect the last segment. In Tao et al. (2001), the authors propose an index, the MV3R-tree, which basically uses both a multi-version R-tree (Becker, Gschwind, Ohler, Seeger & Widmayer, 1996) similar to the HR-tree and a 3D R-tree built on the leaf nodes. The multi-version R-tree is expected to perform better for timeslice or short interval queries, while the 3D R-tree is more adapted for long interval queries. Another interesting structure for indexing the past trajectories of moving objects is described in Pfoser et al. (2000). It still exploits the structure of the R-tree, but tries to group together the segments from the same polyline, which allows to support new types of queries, including the so-called "trajectory queries," with predicates such "enters," "leaves," "crosses," and so forth.

Future trends

As discussed above, so far the proposed index structures fall in one of two categories: either they index the past position, up to the current time; or they index the present and future positions, but their relevancy degrades with time. There is no structure that supports simultaneously both situations, and no fully dynamic index (that is, no index providing an automatic maintenance policy, avoiding periodic, costly re-creation). In spite of the difficulty of the problem, new research efforts are required to address these limitations.

Recently, some specific applications, with constraints that can help to reduce the complexity of the indexing problem, have attracted the attention of researchers. Among them is worth mentioning the common situation of objects moving on a constrained network such as in Pfoser et al. (2003). For instance, the authors propose to index 3D trajectories with two 2D indices, one that contains the network (in the 2D space), and one that contains the transformed trajectories (in 1D for space and 1D for time). Another emerging area of research is the main-memory indexing of moving objects, particularly in the context of moving objects servers providing notification services to customer. In Kalashnikov, Prabhakar, Aref and Hambrusch, (2002), a simple partition of space in cells is used to index the set *S* of moving objects and determine, at each instant, and for each query *q* submitted by a user, the subset of *S* that constitutes the answer to *q*. It is argued that the capacity of computers permits to keep all the structure in main memory, and therefore avoids to design complicated mappings of these structures on disks. More generally, this suggests that emerging Web applications providing services on moving objects raise particular challenges that do not necessarily require the traditional database design approaches.

CONCLUSION

We investigated in this chapter several important issues pertaining to the management of moving objects datasets in databases. The design of representative benchmarks is closely related to the formal characterization of the properties (that is, distribution, speed, nature of movement) of these datasets; uncertainty is another important aspect that conditions the accuracy of the representation and therefore the confidence in query results. Finally, efficient index structures, along with their compatibility with existing software, is a crucial requirement for spatio-temporal databases, as it is for any other kind of data.

The common properties of all the issues considered in this chapter are their strong impact on the representation of data and the way they determine the implementation of both the operations and the data structures that support the evaluation of queries. Indeed, as suggested by the previous discussion, one can envisage many possible applications with quite different features. It is more than likely that the techniques used to manage a database of mobile phone users, a database of cars moving on a road network or a database of airplanes moving freely in a 3D space will strongly or partly differ because of the differents speeds, movement constraints (network-based or not) and behavior. All the aspects (benchmark, uncertainty, indexing) covered, as well as some others (implementation and semantics of database operators, for instance), are affected by these specificities.

We therefore expect in the forthcoming years many other new results, and many improvements to the state-of-the-art solutions that have been established so far.

References

Agarwal, P.K., Arge, L., & Erickson, J. (2000). Indexing Moving Points. *Proceedings of the ACM Symposium on Principles of Database Systems*, 175-186.

Becker, B., Gschwind, S., Ohler, T., Seeger, B., & Widmayer, P. (1996). An Asymptotically Optimal Multiversion B-Tree. *VLDB Journal, 5*(4), 264-275.

Brinkhoff, T. (2000). Generating network-based moving objects. *Proceedings of the International Conference on Scientific and Statistical Databases (SSDBM)*, 253-255.

Brinkhoff, T. (2002). *A framework for generating network-based moving objects*. GeoInformatica, 6(2), 153-180.

Cheng, R., Kalashnikov, D.V., & Prabhakar, S. (2003a). Evaluating probabilistic queries over imprecise data. *Proceedings of ACM SIGMOD International Conference on Management of Data,* 551-562.

Cheng, R., Prabhakar, S., & Kalashnikov, D.V. (2003b). Querying Imprecise Data in Moving Object Environments. *Proceedings of the 19th IEEE International Conference on Data Engineering,*723-725.

Gaede, V., & Guenther, O. (1998). Multidimensional Access Methods. *ACM Computing Surveys, 30*(2), 170-231.

Hadjieleftheriou, M., Kollios, G., Tsotras, V.S., & Gunopoulos, D. (2002). Efficient Indexing of Spatio-temporal Objects. *Proceedings of the International Conference on Extending Data Base Technology*, 251-268.

Kalashnikov, D.V., Prabhakar, S., Aref, W., & Hambrusch, S. (2002). Efficient Evaluation of Continuous Range Queries on Moving Objects. *Proceedings of the International Conference on Databases and Expert System Applications (DEXA)*, 731-740.

Kollios, G., Gunopolos, D., & Tsotras, V.J. (1999). On Indexing Mobile Objects. *Proceedings of the ACM Symposium on Principles of Database Systems*, 261-272.

Moreira, J., Ribeiro, C., & Abdessalem, T. (2000). Query Operations for Moving Objects Database Systems. *Proceedings of the 8th International Symposium on Advances in Geographic Information Systems (ACMGIS-00)*, 108-114.

Moreira, J., Saglio, J.M., & Ribeiro, C. (1999). Representation and manipulation of moving points: An extended data model for location estimation. *Journal of Cartography and Geographic Information Systems, 26*(2), 109-123.

Nascimento, M.A., & Silva, J.R.O. (1998). Towards historical r-trees. *Proceedings of the ACM International Symposium on Applied Computing,* 235-240.

Nascimento, M.A., Silva, J.R.O., & Theodoridis, Y. (1999). Evaluation for Access Structures for Discretely Moving Points. *International Workshop on Spatio-Temporal Database Management (STDBM'99),* LNCS 1678, 171-181.

Pfoser, D., & Jensen, C.S. (1999). Capturing the uncertainty of moving object representations. *Computer Science, 1651,* 111-132.

Pfoser, D., & Jensen, C.S. (2003). Indexing of Network-Constrained Moving Objects. *Proceedings of the International Symposium on Geographic Information Systems,* 25-32.

Pfoser, D., Jensen, C.S, & Theodoridis, Y. (2000). Novel Approaches in Query Processing for Moving Objects. *Proceedings of the International Conference on Very Large Data Bases (VLDB),* 395-406.

Pfoser, D., & Tryfona, N. (2001). Capturing fuzziness and uncertainty of spatiotemporal objects. *Computer Science, 2151,* 112.

Porkaew, K., Lasaridis, I., & Mehrotta, S. (2001). Querying Mobile Objects in SpatioTemporal Databases. Proceedings of the International Symposium on Spatial and Temporal Databases (SSTD), 59-78.

Rigaux, P., Scholl, M., & Voisard, A. (2001). *Spatial Databases.* Morgan Kaufmann.

Saglio, J.M., & Moreira, J. (2001). Oporto: a realistic scenario generator for moving objects. *GeoInformatica, 5*(1), 71-93.

Saltenis, S., Jensen, C.S., Leutenegger, S.T., & Lopez, M.A. (2000). Indexing the Positions of Continuously Moving Objects. *Proceedings of the ACM SIGMOD Symposium on the Management of Data,* 331-342.

Saltenis, S., & Jensen, C.S. (2002). Indexing of Moving Objects for Location-Based Services. *Proceedings of the IEEE International Conference on Data Engineering (ICDE),* 463-472.

Sistla, P., Wolfson, O., Chamberlain, & Dao, S. (1998). Querying the uncertain position of moving objects. *Computer Science, 1399,* 310.

Tao, Y., & Papadias, D. (2001). The MV3R-Tree: a Spatial-Temporal Access Method for Timestamp and Interval Queries. *Proceedings of the International Conference on Very Large Data Bases (VLDB),* 431-440.

Tao, Y., Papadias, D., & Sun, J. (2003). The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. *Proceedings of the International Conference on Very Large Data Bases (VLDB),* 790-801.

Tayeb, J., Ulusoy, O., & Wolfson, O. (1998). A Quadtree Based Dynamic Attribute Indexing Method. *Computer Journal*, 41, 185-200.

Theodoridis, T., Silva, J.R.O., & Nascimento, M.A. (1999a). On the Generation of Spatiotemporal Datasets. *Computer Science, 1651,* 147-164.

Theodoridis, T., Silva, J.R.O., & Nascimento, M.A. (1999b). On the Generation of Spatiotemporal Datasets. *Proceedings of the International Conference on Large Spatial Databases (SSD'99),* 147-164.

Theodoridis, Y. (2003). Ten Benchmark Database Queries for Location-based Services. *Computer Journal*, *46*(6), 713-725.

Theodoridis, Y., Vazirgiannis, M, & Sellis, T. (1996). Spatio-temporal Indexing for Large Multimedia Applications. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems,* 441-448.

Trajcevski, G. (2003). Probabilistic range queries in moving objects databases with uncertainty. *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 39-45.

Trajcevski, G., Wolfson, O., Zhang, F., & Chamberlain, S. (2002). The geometry of uncertainty in moving objects databases. *Proceedings of the 8th International Conference on Extending Database Technology,* LNCS, vol. 2287, 233-250.

Wolfson, O., Jiang, L., Sistla, A.P., Chamberlain, S., Rishe, N., & Deng, M. (1999a). Databases for tracking mobile units in real time. *Computer Science*, *1540*, 169-186.

Wolfson, O., Sistla, A.P., Chamberlain, S., & Yesha, Y. (1999b). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, *7*(3), 257-387.

Wolfson, O., Sistla, A.P., Xu, B., Zhou, J., Chamberlain, S., Yesha, Y., & Rishe, N. (1999c). Tracking moving objects using database technology in DOMINO. *Next Generation Information Technologies and Systems,* 112-119.