Bulletin of the Technical Committee on

Data Engineering

June 2010 Vol. 33 No. 2

IEEE Computer Society

Letters

Special Issue on Spatial and Spatio-temporal Databases

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TC on Data Engineering web page is
http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at
http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

Bulletin Editors

I told you in the March issue about prior editors who were retiring. Let me now take this opportunity to introduce our new editors, whose information now appears on the inside cover of this June Bulletin issue. The new editors, in alphabetic order are (1) Peter Boncz of CWI in the Netherlands, Brian Cooper of Yahoo! in California, Mohamed Mokbel of the University of Minnesota, and Wang-Chiew Tan of IBM in California. As is always the case I strive to find editors who are leading researchers whose collective interests and expertise span areas of current interest to the database community. I am confident that the new editors will continue to bring the latest research and technology issues, trends, and successes, whether from academic or industrial organizations, to Bulletin readers. It is a pleasure to once again find terrific researchers who are willing to take on this task, and I look forward to working with them over the next two years.

TCDE Chair Election

There is a separate letter on page two of this issue calling for nominations for the position of TCDE Chair. In the past, participation has been low. I would urge you, however, to give some thought to whom you would like to see in this position, and let us know by nominating him or her. This can be done by an email to any of the nominating committee members.

The Current Issue

Spatial and spatio-temporal databases have become increasingly important as a research topic. This is in no small part due to the commercial opportunity in providing services to people based on where they are and, e.g., the time of day. Thus there has been a flurry of commercial activity in this area. This interest has been sufficient so that there is now a SIGSPATIAL group within the ACM that is sponsoring its own SIGSPATIAL conference. We are fortunate in having the Co-PC chair of this year's SIGSPATIAL conference, Mohamed Mokbel, as a Bulletin editor. Mohamed has assembled the current issue, exploiting his well-informed knowledge of what is happening in this area and who is doing it. The result is an issue that encompasses a broad cross section of the research and industrial activity in this area. I am sure you will be well rewarded in reading this issue. Perhaps it will convince you to also pursue this as a research area yourself.

I want to thank Mohamed for his efforts in assembling the issue. It contains work from both academic and industrial people, continuing a Bulletin strength in describing technology to our readers from across the entire technology community. Additional thanks are due Mohamed for his willingness to immediately take on an issue upon his being appointed as editor.

David Lomet
Microsoft Corporation

Letter Calling for TCDE Chair Nominations

Calling for TCDE Chair Nominations

The Chair of the IEEE Computer Society Technical Committee on Data Engineering (TCDE) is elected for a two-year period. The current Chair, Paul Larson, is unavailable to be a candidate. A Nominating Committee for electing a new chair for the period 2011-2012 consisting of Calton Pu, David Lomet and Erich Neuhold has been formed. The Nominating Committee invites nominations for the position of Chair from all members of the TCDE. To submit a nomination, please contact any member of the Nominating Committee before August 25, 2006.

More information about TCDE can be found at
`http://tab.computer.org/tcde/index.html`.

Information about TC elections can be found at
`http://www.computer.org/tab/hnbk/electionprocedures.htm`.

Erich Neuhold, Calton Pu, David Lomet
TCDE Nominating Committee

Letter from the Special Issue Editor

Spatial and spatio-temporal databases provide backbone support for a set of widely used applications including geographic information systems, location-based services, moving objects databases, transportation, and emergency services. This special issue includes ten articles geared towards new frontiers of spatial and spatio-temporal databases.

The first article by Sankaranarayanan and Samet presents a paradigm shift in querying road networks where they strongly advocate for storing road networks in relational databases, as opposed to the widely used graph data structure. The article introduces a new data structure, called road network oracle, that resides in a database and enables the processing of many operations on road networks with just the aid of relational operators. Doing so also takes advantage of the power of SQL queries along with the database query optimizers. The second article by Jensen *et al.* presents one of the first attempts to efficiently track moving objects in indoor environments. In contrast to the commonly used outdoor environments, indoor environments suffer from inaccurate positioning and complex topologies.

The third article by Wolfson and Xu describes very interesting applications, research issues, and approaches related to applying spatio-temporal databases in urban transportation, from trip planning and navigation, to abstraction of concepts from spatio-temporal sensor data, mobile peer-to-peer environments, and social networks. The fourth article by Zhou *et al.* presents the application of spatio-temporal databases in emergency situations. More specifically, the article presents an efficient approach for evacuation planning in case of natural disasters or terrorist attacks. Getting a lot of media attention (e.g., Fox TV News), this article presents one of the very unique applications of spatio-temporal databases.

The following two articles address the use of spatio-temporal data in social networks. Banaei-Kashani *et al.* present GeoSIM (Geospatial Social Image Mapping), a system that enables a group of users with camera-equipped mobile phones to participate in a collaborative collection of urban texture information. GeoSIM has the ability to enable inexpensive, scalable, and high resolution data collection of urban texture mapping. Zheng *et al.* introduce GeoLife, a social networking service which aims to understand and mine trajectories, locations, and users. GeoLife also aims to share life experience among its users based on their GPS trajectories and provide personalized friend, travel, and location recommendations to its users.

The seventh article by Nguyen-Dinh *et al.* gives a very thorough survey of a large number of spatio-temporal access methods, widely used to support the various spatio-temporal applications discussed in this issue. This survey focuses on access methods developed since 2003, where an earlier version of the survey covering access methods up to 2003 was published in this bulletin on June 2003. The eighth article by Güting *et al.* presents the SECONDO extensible database system into which a lot of moving object technology has been built already. SECONDO is one of the unique open-source systems built in academia that allows researchers to implement their new techniques in moving object databases within a system context and to make them available for practical use by other researchers. SECONDO has the ability to transform research in spatio-temporal databases into a new frontier where system-oriented research can take place, which would have significant impact on industry.

The issue is then concluded by two industrial articles from IBM and Microsoft that discuss supporting spatio-temporal data streams using IBM System S and Microsoft StreamInsight, respectively. Both articles present the first *industrial* attempts to support spatio-temporal data streams. Biem *et al.* utilize IBM System S to support real-time traffic information management in the city of Stockholm where large volumes of GPS data are collected and processed online. Ali *et al.* discusse the native support of spatio-temporal streams in Microsoft StreamInsight along with the ongoing effort at Microsoft SQL Server to bring together the temporal aspect of StreamInsight and the spatial support of the SQL Server Spatial library.

Mohamed F. Mokbel
Department of Computer Science and Engineering,
University of Minnesota
Minneapolis, MN, USA

Roads Belong in Databases

Jagan Sankaranarayanan Hanan Samet
Center for Automation Research
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland, College Park, MD 20742
$\{jagan, hjs\}$@$cs.umd.edu$

Abstract

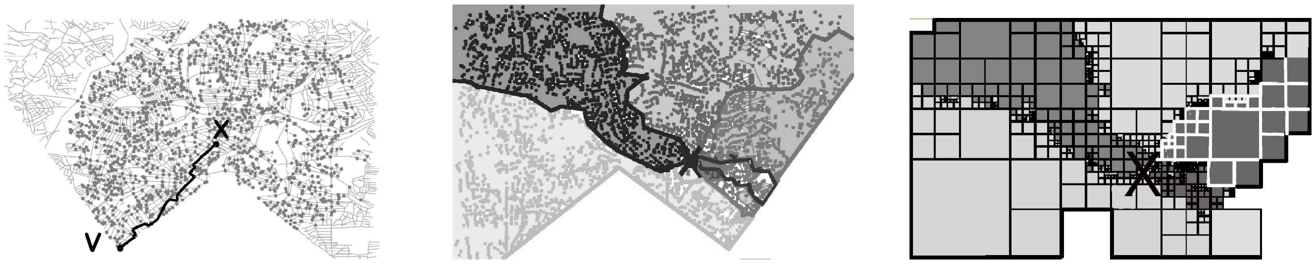
The popularity of location-based services and the need to perform real-time processing on them has led to an interest in queries on road networks, such as finding shortest paths and finding nearest neighbors. The challenge here is that the efficient execution of operations usually involves the computation of distance along a spatial network instead of "as the crow flies," which is not simple. This requires the precomputation of the shortest paths and network distance between every pair of points (i.e., vertices) with as little space as possible rather than having to store the n^2 *shortest paths and distances between all pairs. This problem is related to a 'holy grail' problem in databases of how to incorporate road networks into relational databases. A data structure called a road network oracle is introduced that resides in a database and enables the processing of many operations on road networks with just the aid of relational operators. Two implementations of road network oracles are presented.*

1 Introduction

The growing popularity of online mapping services such as Google Maps, Microsoft Bing, and Yahoo! maps has led to an interest in responding to queries in real time, such as finding shortest routes between locations along a spatial network as well as finding nearest objects from a set S (e.g., gas stations, markets, and restaurants) where the distance is measured along the shortest path in the network. Elements of S are usually constrained to lie on the network or at the minimum to be easily accessible from the network. The online nature of these services means that responses must be generated in real time. The challenge in performing queries on road networks is that operations involve the computation of distance along a spatial network (i.e., network distance) instead of "as the crow flies," which is not simple.

Operations on road networks [3, 4, 12, 22, 13, 14, 16, 18, 17, 15, 19, 21, 20] are expensive because computing distances between two objects (e.g., postal addresses) on the road network requires the invocation of a shortest path algorithm [5, 8, 11, 24, 1, 6]. A popular shortest path algorithm is Dijkstra's algorithm [5], which if invoked between a source vertex q and a destination vertex v, ends up visiting every vertex that is closer to q via the shortest path from q than v.In particular, it is not uncommon for Dijkstra's algorithm to visit a very large number of the vertices of the network in the process of finding the shortest path between vertices that are

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering



(a) (b) (c)

Figure 1: (a) Map of Silver Spring, MD where the highlighted vertices are those visited by Dijkstra's algorithm in determining the shortest path from X to V, (b) partition into regions r_i such that the shortest path from X to a vertex in r_i passes through the same vertex among the six vertices adjacent to X (i.e., the shortest-path map of X), and (c) leaf blocks in the shortest-path quadtree for the regions of the same partition.

reasonably far from each other in terms of network hops. For example, Figure 1(a) shows the vertices that would be visited when finding the shortest path from the vertex marked by X to the vertex marked by V in a spatial network corresponding to Silver Spring, MD. Here we see that in the process of obtaining the shortest path from X to V of length 75 edges, 75.4% of the vertices in the network are visited (i.e., 3,191 out of a total of 4,233 vertices).

Methods such as the *transit node routing* of Bast et al. [1] and the landmark approach of Goldberg and Harrelson [8] as well as [6] can significantly speedup shortest path computations on large road networks, up to several orders of magnitude compared to Dijkstra's algorithm. In spite of these newer approaches being much faster than Dijkstra's algorithm, they still involve searches on graphs during runtime. That is, when the user is waiting for an answer, network distance computation still involves searching a graph space denoted by the road network for a shortest path so that network distances can be obtained. Such a model for performing operations on road networks is unsuitable for building a real-time large scale system because of the computational complexity of computing network distances in road networks.

Therefore, it is not surprising that web services allow users to compute shortest paths, but if presented with any task that is a bit more complicated (such as finding k nearest neighbors), then they resort to using the Euclidean distance (i.e., as the crow flies); but the error due to this approximation is generally unacceptable. If operations on road networks must compute the network distances on the fly, then requiring that the result be obtained in real time (or almost real time) precludes the use of conventional algorithms that are graph-based (e.g., nearest neighbor finding methods such as the INE and IER methods [13] and improvements on them [4]), which usually incorporate Dijkstra's algorithm [5] in at least some parts of the solution [14]. We suggest precomputing of the shortest paths between vertices in a road network as the only way of building scalable services that perform real-time operations on road networks.

Road Networks in Databases: A source of frustration for database researchers with road networks is that they cannot be integrated easily into a relational database system. Understanding why this is important requires a bit more explanation. When we think of road networks, we view them as general graphs as there is an easy transformation from a road network to a general graph equivalent G. The transformation is achieved by casting road intersections as vertices, road segments as edges, and distances in miles or time taken to travel a road segment in the road network as edge weights in G. Furthermore, additional constraints on road networks such as one ways, no left hand turns etc., can be suitably handled by including directions with edges, or making small changes to the topology of the graph equivalent G. Now, operations on road networks are cast as combinatorial operations (i.e., graph operations) on G. From a database point of view, casting operations on road networks as graph operations on G is not a good strategy. In particular, combinatorial operations on G cannot be cast in terms of relational operators (i.e., select, project, join etc.), which constitute the basic operators of any relational database system. As a consequence, operations on road networks cannot be performed in the context of a database system, which has the unfortunate implied consequence that operations on road networks cannot be expressed using SQL.

Now, why is expressing operations on road networks using SQL important? Relational database systems allow operations to be written in SQL, which is an English-like language. SQL is quite expressive which means that people can build applications quickly, without having to concern themselves with how the queries are actually processed. This translates easily for a database system as queries written in SQL can be easily optimized by a query optimizer, since any query in SQL can be rewritten exclusively in terms of relational operators. Given that the database knows how these few operators work (also maintain statistics to aid them), it is fairly straightforward to optimize queries written in SQL. Being able to express operations using SQL is especially critical to road networks given that many branches of science and engineering deal with road networks and run expensive operations on them.

We argue that integrating road networks into a database makes good sense from a systems point of view. Suppose that we want to develop an application that finds all restaurants within 10 miles of a given postal address. If we were to design this algorithm, then we would store the restaurants as a relation in a database system. When a query point q is given, we would have to first query the database to obtain the set of restaurants that are likely to be within 10 miles of q, process this query in an external graph processing module, and then possibly store the result in the database. It would make for a much cleaner system design if we could perform all the operations in the context of a database system, which is really our goal here.

In Defense of Precomputation: As we pointed out earlier, integrating road networks into a relational database, requires a reassessment of how we deal with them. In this context, precomputation gives us an opening to transform road networks from a graph data structure to an alternate representation that can potentially make redundant the "graph" (or topological) aspect of most operations on road networks. The key idea is that by precomputing the shortest paths and distances between every pair of vertices in a road network, we can build a data structure residing in a database that can perform operations on road networks using SQL.

The question that we explore next is whether precomputation a bad strategy. We argue that it is not, pointing out that although the precomputation task is a big computational problem, it is not impossibly large. Of course, there is no denying that precomputation can be massively expensive for large road networks, but it can be achieved with a sufficient investment of time and hardware resources. We first argue that such a representation is feasible to compute. In particular, given a graph $G(V, E)$, where $n = |V|$, and $m = |E|$, Dijkstra's algorithm using a Fibonacci heap [7] takes $O(n^2 \log n + nm)$ time to compute the shortest path between all pairs of vertices in a spatial network. When $m = O(n)$, as in road networks, the time complexity of Dijkstra's algorithm would be $O(n^2 \log n)$. Empirical studies [25] have indicated that Dijkstra's algorithm is no where close to being the fastest algorithm for computing the all-pairs shortest paths on road networks. Moreover, recent developments in the shortest paths algorithm literature have shown better theoretical bounds on the computational time. In particular, Henzinger *et al.* [10] present a linear time shortest path algorithm for planar graphs, while Thorup [23] provides a linear time shortest path algorithm for general graphs with integer edge weights. Moreover, a host of other techniques such as parallel processing and the use of sophisticated hardware such as Graphics Processing Units (GPU) [9] could further speed up the precomputation of all shortest paths of a graph.

The real problem with precomputation is that the resulting precomputed representation is an impossibly large. Consider a spatial network $G(V, E)$ containing n vertices. Storing the n^2 shortest paths can require $O(n^3)$ space, when we assume that each shortest path can potentially have $O(n)$ vertices. Given that n is around 24 million vertices for the road network of USA, the anticipated size of the shortest path representation is around 13 billion trillion pieces of information, which is indeed impossibly large to store. Actually, one immediate reduction in the storage requirement can be achieved by just storing an intermediate vertex w ("next hop") on the shortest path from source s to destination t. We can then obtain the entire shortest path between s and t, by repeatedly finding the next vertex in the shortest path between s and w, and w and t, and so on. This reduces the total amount of space necessary to $O(n^2)$, which comes at the cost of making shortest path retrieval into an iterative process taking $O(k)$ time, where k is the length of the shortest path. Even such a representation is quite large requiring 576 trillion pieces of storage for the road network of USA. In this article, we show that we can substantially reduce this precomputed input into a much compact representation that is just linear in n [19, 21].

It is important to note that precomputation is actually a good idea arguing from the point of view of *decoupling*, which provides a cost justification to any precomputation strategy. In particular, suppose that we precompute and store the shortest path and network distance between every pair of vertices in the road network of Manhattan, NY. Such a representation can potentially be used by several datasets (possibly, millions of user generated datasets) pertaining to postal addresses in Manhattan for query processing. Moreover, road networks are usually static structures, while datasets of objects may be updated frequently. For example, when dealing with a set of mobile hosts on a road network, the current positions of the objects are frequently updated, while the road network would largely remain static. So, precomputation is largely a one-time affair. In effect, precomputation enables us to *decouple* the *data* from the *underlying domain* which allows for the datasets and the network representation to be largely independent of each other.

2 Road Network Oracle

Our precomputed input representation of a road network containing n vertices is of size $O(n^3)$. Our goal here is to convert it to a database friendly representation that is much smaller in size. The Road Network Oracle (or simply *Oracle*) O of a road network G is a data structure that completely encapsulates all the n^2 shortest paths and network distances between every pair of vertices in G. The obvious storage choice for O in a relational database system is a *database relation*. In the following, we provide a basic blueprint for an oracle without making too many assumptions. Later in Section 3, we show how we can construct oracles for road networks.

The oracle O allows for two operations at the minimum, but of course support for additional functionalities is clearly preferable. Given a source vertex u and a destination vertex v, O provides an intermediate vertex in the shortest path between u and v. Next, given u and v, O also provides the network distance between u and v or more likely an approximation of it. Now, the key restriction that we place on O is that its size should be small so that both these operations can be be performed in a reasonable amount of time using database indices in O.

Without loss of generality, the scheme of oracle O can be deduced by observing that the input representation contains $O(n^2)$ information, which consists of an intermediate vertex in the shortest path between every pair of vertices, as well as the network distances between them. Of course, storing all of this information is not an option here, which means that we need to find a way to obtain a compact equivalent representation by removing *redundancies* from the input. One way to do this by aggregating a set of source vertices A and a set of destination vertices in B, such that the shortest paths from a source vertex in A to a destination vertex in B share a common vertex Ψ. Now, instead of recording the same intermediate vertex Ψ for every pair of source and destination vertices in A to B, respectively, we can simply represent all the individual vertices in A by their group identity (i.e., A), and all the individual vertices in B by their group identity (i.e., B). Similarly, network distances between source vertices in A to destination vertices in B, that are more or less of the same value can also be succinctly captured by just storing just a single network distance for the shortest paths from vertices in A to B.

The schema of a relation that captures the shortest paths is given by: $O(AB, \Psi)$, where AB represents the group identity of the vertices in the road network belonging to A and B, such that Ψ is the common intermediate vertex on the shortest paths between them. Another relation records the network distances separately using another relation with the schema: $O(AB, d_{apx})$, where again AB is the group identity of the source and destination vertices whose network distances are approximated with d_{apx}. Of course, for the sake of simplicity, if we assume (not without merit) that the vertices that make up AB in the relation storing the shortest paths are also the same ones that make up AB in the relation storing the network distances, then we can combine these two relations into a single relation with the schema: $O(AB, \Psi, d_{apx})$. The oracle O should capture all the $O(n^2)$ shortest paths and network distances. Given any pair of vertices p and q in the road networks, there should be exactly one tuple in O such that AB contains p and q. However, the groups AB should be maximal so that the number of tuples in the schema should be at a minimum. Finally, the group AB to which the pair (p, q) belongs should be unique so that it can be found quickly with the aid of the B-tree on the attribute AB in O.

Operations on Road Networks using SQL Given an implementation of $O(AB, \Psi, d_{apx})$, we now show how we can efficiently perform query processing on it using SQL. In particular, we demonstrate how many operations on road networks can be expressed using SQL (and relational operators) in the context of a database system. Our example assumes the following setup. Let R be a relation of restaurants with schema *(id, type, price)*, where *id* uniquely identifies restaurants on the road network, *type* is the kind of the cuisine served by the restaurant, and *price* is the average cost. We also define another relation Q of movie theaters given by the same schema *(id, movie_id)* where *id* uniquely identifies the movie theater on the road network, and *movie_id* is a movie playing in the theater. Given a source p and destination q, let $Z(p, q)$ map them to a group key AB such that A contains p, and B contains q so that one can find a tuple in O with the aid of B-tree on $O.AB$ such that the value of AB equals $Z(p, q)$. We present the following queries on a spatial network.

Approximate Network distance: Given a source p and destination q, obtain an approximate network distance between them.

SELECT $O.d_{apx}$ FROM O WHERE $O.AB = Z(p, q)$

Region Search: Given a query location q, obtain all restaurants in R that are within 10 miles of q which serve *Italian* cuisine. Of course, this query would make more sense if the oracle can give ϵ-guarantees on the quality of the network distance answers.

SELECT R.id, $O.d_{apx}$ FROM R, O WHERE $O.AB = Z(q, R.\text{id})$ and R.type = "Italian" and $O.d_{apx} \leq 10$ miles

k-Nearest Neighbor Search: Given a query location q, determine the k closest restaurants in R to q which serve *Italian* cuisine.

SELECT R.id, $O.d_{apx}$ FROM R, O WHERE $O.AB = Z(q, R.\text{id})$ and R.type = "Italian" ORDER BY $O.d_{apx}$ LIMIT k

Distance Join: Find the k closest pairs of restaurants in R and movie theaters in Q, such that Q is playing a movie of movie_id m.

SELECT R.id, Q.id, $O.d_{apx}$ FROM R, Q, O WHERE $O.AB = Z(R.\text{id}, Q.\text{id})$ AND Q.movie_id = m ORDER BY $O.d_{apx}$ LIMIT k

For a discussion on how the query optimizer can optimize queries on O, the reader is referred to [20].

3 Oracle Implementations

Now that we have laid out the design of an oracle, we can try to implement a few oracles. First of all, the input $O(n^2)$ can be stored as an oracle $O(uv, \Psi, d_G(u, v))$, where u is a source vertex, v is a destination vertex such that Ψ is an intermediate vertex in the shortest path from u to v and $d_G(u, v)$ is the network distance between u and v. The main drawback of this oracle is that it contains $O(n^2)$ tuples, which renders it infeasible owing to its large size.

We now present the design of two oracles both of which are based on the observation that road networks also contain additional information in the form of the spatial positions of road intersections (vertices) and road segments (edges). Shortest paths are related to the spatial information by the observation that the shortest paths

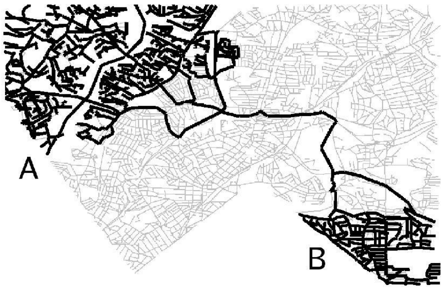


Figure 2: Example illustrating the path coherence in a road network of Silver Spring, MD. The 30,000 shortest paths (given in a darker shade) between every pair of vertices in A and B pass through a single vertex.

between spatially proximate source vertices and spatially proximate destination vertices will often pass through a common vertex, which is termed *path coherence*. Moreover, when the proximate source vertices are far from the proximate destination vertices we have the situation that the shortest paths between them have many common segments.

Figure 1(b) shows a colored map obtained by applying a coloring algorithm to the vertices based on the shortest path from a source vertex u in the road network G of Silver Spring, MD. Now, each vertex in G (other than u) is assigned a color based on which outgoing edge of u forms the first link in the shortest path from u. For example, u has six outgoing edges and hence, every vertex in G is assigned one of the 6 possible colors based on which of the 6 outgoing edges of u form the first link in the shortest path from u. We see that the resulting colored map has contiguous colored regions, which indicates that two destination vertices v, w in G that are close to one another, but spatially far from u share common segments in the shortest path from u.

An oracle, termed the *SILC Oracle*, which captures the above scenario is obtained as follows. Given a source vertex q, we perform such a coloring operation on all the vertices in the road network so that all the vertices except q have a unique color. Next, we can construct a quadtree representation as shown in Figure 1(c) on the colored vertices that keeps subdividing a block until all the vertices are of the same color, which means that they share the same first edge from q. We record the identity of these blocks using a pointerless quadtree representation which in essence is a pair of numbers where one number corresponds to the path from the root of the tree to the block and the other number corresponds to the depth of the block. These two numbers are concatenated to form an integer and the result is known as the block's *Morton block representation*. Let B represent one of these Morton blocks. Next, the shortest-path quadtree of a vertex is stored in a relation $O(q, B, \Psi, d_{apx})$, where q corresponds to a source vertex, B is the Morton block representation of one of the blocks in the quadtree, Ψ is the first link in the shortest path from q to the vertices contained in B, and d_{apx} is a network distance that approximates the network distances from q to all the vertices in B. We compute the shortest-path quadtree for each vertex in the road network and store the tuples in O. The shortest-path quadtree reduces the storage requirements from $O(n^2)$ to $O(n^{1.5})$ [15]. O is indexed using a two-dimensional B-tree on q, B. Given a source vertex u and a destination vertex v, O can retrieve the intermediate vertex and the approximate network distance in $O(\log n)$ time. The only drawback of this oracle is that d_{apx} is not of a bounded approximation, which means that for some cases the errors can be large.

The next oracle, termed the *path-distance oracle* [21, 19], is obtained by utilizing the path coherence between sets of source and destination vertices such that they share common vertices in the shortest paths between them. Figure 2 shows a configuration of source vertices A and destination vertices B such that every shortest path from a vertex $u \in A$ to a vertex $t \in B$ passes through a particular set of vertices, resulting in storing partial path information of 30,000 shortest paths using $O(1)$ storage. The triple (A, B, u) is said to form a *Path-Coherent Pair (PCP)* if and only if all the shortest paths from source vertices in A to destination vertices in B have at least one vertex or one edge in common u as shown in Figure 2. In particular, we can show [19] that a given road network of size n, can be broken into $O(n)$ such groups of $O(1)$ size that capture all of the $O(n^2)$ shortest paths of the network. This partitioning of the vertices into appropriate subsets of source and destination vertices is achieved by appealing to the Well-Separated Pair (WSP) decomposition [2], and conditions under which it is satisfied for a spatial network are specified in [21]. Moreover, as the shortest paths between A and B share large segments, the network distances between source vertices in A and destination vertices in B can be approximated by a single network distance value d_ϵ that provides ϵ-guarantees on the quality of the answers [19, 21, 20]. The end result is that given a road network G containing n vertices, we can break it up into $O(s^2n)$ sets (s is a small value > 0) of the form (AB, Ψ, d_ϵ) such that AB encapsulates the set of source and destination vertices, Ψ is an intermediate vertex common to all the shortest paths from A to B, and d_ϵ approximates all the network distances between A and B. This oracle satisfies all the conditions that we laid out earlier in Section 2 as it is linear in size, with no redundancy and can retrieve both the intermediate vertex and ϵ-approximate network distance quickly.

4 Concluding Remarks

In this paper we presented an alternate way of performing operations on road networks using road network oracles, which reside in a database system and enable operations on road networks using SQL. Our approach of converting road networks into oracles provides an opportunity to move away from traditional methods of working with road networks towards a way that is scalable, efficient, database friendly, and being able to support Internet-scale real time operations. We presented a few possible implementations of the oracle that use the spatial information in a road network to provide group memberships to vertices in the road network. The path-distance oracle [19, 20, 21] can be shown to be linear in n as well as being able to provide an intermediate vertex in the shortest path as well as an ϵ-approximate network distance between any pair of vertices drawn from the road network [21]. An interesting challenge is exploring whether there are other schemes of providing vertices in a road network with group memberships (á la spatial information) so that given source and destination vertices, one can find a tuple containing the pertinent shortest path and distance information using a B-tree. Another open problem is whether dynamic oracles can be designed so that they can deal with updates as is the case when road segments or vertices become closed or road segments become one way, as well as other dynamic traffic conditions such as congestion which may make some routes more acceptable. Finally, there is the issue of how to optimize operations involving the oracles so that operations can be optimized effectively in the context of a relational database system. In particular, how can a query optimizer be taught more strategies to perform road network queries more effectively.

Acknowledgments: This work was supported in part by the National Science Foundation under Grants IIS-09-48548, IIS-08-12377, CCF-08-30618, and IIS-07-13501, as well as NVIDIA Corporation, Microsoft Research, Google, the E.T.S. Walton Visitor Award of the Science Foundation of Ireland, and the National Center for Geocomputation at the National University of Ireland at Maynooth.

References

[1] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes. In transit to constant time shortest-path queries in road networks. In *ALENEX*, pp. 34–43, New Orleans, LA, Jan 2007.

[2] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *SODA*, pp. 291–300, Austin, TX, Jan. 1993.

[3] Z. Chen, H.-T. Shen, X. Zhou, and J. X. Yu. Monitoring path nearest neighbor in road networks. In *SIGMOD*, pp. 591–602, Providence, RI, June 2009.

[4] H.-J. Cho and C.-W. Chung. An efficient and scalable approach to CNN queries in a road network. In *VLDB*, pp. 865–876, Trondheim, Norway, Sep. 2005.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[6] G. G. Filho and H. Samet. A hybrid shortest path algorithm for intra-regional queries in hierarchical shortest path finding. CS Tech. Report TR–4417, University of Maryland, College Park, MD, Nov. 2002.

[7] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *JACM*, 34(3):596–615, July 1987.

[8] A. V. Goldberg and R. F. Werneck. Computing point-to-point shortest paths from external memory. In *ALENEX*, Vancouver, Canada, January 2005.

[9] P. Harish and P. J. Narayanan. Accelerating large graph algorithms on the GPU using CUDA. In *HiPC*, vol. 4873 of LNCS, pp. 197–208, Goa, India, Dec. 2007.

[10] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *JCSS*, 55(1):3–23, Aug. 1997.

[11] N. Jing, Y.-W. Huang, and E. A. Rundensteiner. Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation. *TKDE*, 10(3):409–432, May 1998.

[12] M. R. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*, pp. 840–851, Toronto, Canada, Sep. 2004.

[13] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, pp. 802–813, Berlin, Germany, Sep. 2003.

[14] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006.

[15] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD*, pp. 43–54, Vancouver, Canada, June 2008.

[16] J. Sankaranarayanan, H. Alborzi, and H. Samet. Efficient query processing on spatial networks. In *ACM GIS*, pp. 200–209, Bremen, Germany, Nov. 2005.

[17] J. Sankaranarayanan, H. Alborzi, and H. Samet. Distance join queries on spatial networks. In *ACM GIS*, pp. 211–218, Arlington, VA, Nov. 2006.

[18] J. Sankaranarayanan, H. Alborzi, and H. Samet. Enabling query processing on spatial networks. In *ICDE*, pp. 163, Atlanta, GA, Apr. 2006.

[19] J. Sankaranarayanan and H. Samet. Distance oracles for spatial networks. In *ICDE*, pp. 652–663, Shanghai, China, Apr. 2009.

[20] J. Sankaranarayanan and H. Samet. Query processing using distance oracles for spatial networks. *TKDE*, 2010. Best Papers of ICDE 2009 Special Issue. To appear.

[21] J. Sankaranarayanan, H. Samet, and H. Alborzi. Path oracles for spatial networks. In *VLDB*, volume 2, pp. 1210–1221, Lyon, France, Aug. 2009.

[22] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica*, 7(3):255–273, Sep. 2003.

[23] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *JACM*, 46(3):362–394, May 1999.

[24] D. Wagner and T. Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *ESA 2003*, vol. 2832 of LNCS, pp. 776–787, Budapest, Hungary, Sep. 2003.

[25] F. Zhan and C. E. Noon. Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32(1):65–73, Feb. 1998.

Indoor—A New Data Management Frontier

Christian S. Jensen† Hua Lu† Bin Yang‡

†Department of Computer Science, Aalborg University, Denmark

{csj, luhua}@cs.aau.dk

‡School of Computer Science, Fudan University, China

byang@fudan.edu.cn

Abstract

Much research has been conducted on the management of outdoor moving objects. In contrast, relatively little research has been conducted on indoor moving objects. The indoor setting differs from outdoor settings in important ways, including the following two. First, indoor spaces exhibit complex topologies. They are composed of entities that are unique to indoor settings, e.g., rooms and hallways that are connected by doors. As a result, conventional Euclidean distance and spatial network distance are inapplicable in indoor spaces. Second, accurate, GPS-like positioning is typically unavailable in indoor spaces. Rather, positioning is achieved through the use of technologies such as Bluetooth, Infrared, RFID, or Wi-Fi. This typically results in much less reliable and accurate positioning.

This paper covers some preliminary research that explicitly targets an indoor setting. Specifically, we describe a graph-based model that enables the effective and efficient tracking of indoor objects using proximity-based positioning technologies like RFID and Bluetooth. Furthermore, we categorize objects according to their position-related states, present an on-line hash-based object indexing scheme, and conduct an uncertainty analysis for indoor objects. We end by identifying several interesting and important directions for future research.

1 Introduction

During primarily the past decade, an increasingly large body of research results on moving objects has come into existence (e.g., [1, 6, 10, 12, 11]). Some of these results serve as a technology foundation for the growing location-based services (LBSs) industry. However, most moving-object research to date assumes an outdoor setting with GPS, or GPS-like, positioning. This research, unfortunately, falls short in another very important setting, namely indoor spaces.

Indoor spaces may accommodate very large populations of moving individuals. In fact, people spend large parts of their lives in indoor spaces such as private homes, office buildings, shopping malls, conference facilities, airports, and subway stations. With positioning being available in indoor spaces, it is easy to imagine that we are able to provide a wide range of indoor location-based services akin to those enabled by GPS-based positioning in outdoor settings. Example indoor services include navigation, personal security, a variety of location-based information services, and services providing insight into how and how much an indoor space is being used.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Much of the research on outdoor moving objects is not easily applicable in indoor settings. This can be attributed in part to two differences between indoor and outdoor settings.

First, indoor spaces are composed of entities that are unique to indoor settings: often rooms and hallways connected by doors, as exemplified in Figure 1.

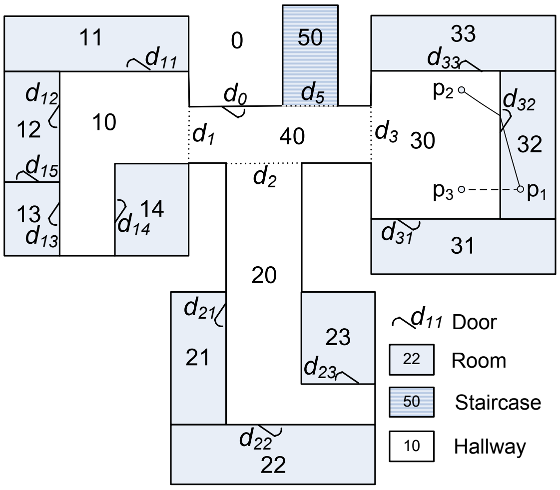


Figure 1: Example Indoor Space

These entities enable and constrain movement. In the example, a user who wishes to move from location p_1 to location p_3 must to go through door d_{32}; the wall between room 32 and room 30 blocks the direct movement.

Such constraints render the conventional Euclidean distance inapplicable in indoor settings. If disregarding the indoor topology, location p_1's (Euclidean) nearest neighbor is p_3. However, taking the indoor topology into consideration, p_1's true nearest neighbor is p_2.

In addition, indoor movement is less constrained than outdoor spatial-network constrained movement, where the position of an object is constrained to a position on a polyline. Consequently, symbolic models rather than geometric models are often used for modeling indoor spaces [3].

Second, GPS-like positioning is typically unavailable in indoor spaces. Rather, other positioning technologies are deployed in indoor settings that differ fundamentally from GPS-like positioning. Specifically, technologies that have been proposed for short-range communication, such as RFID [13], Bluetooth [4], and Infrared, can be exploited for indoor positioning. However, unlike GPS that is able to report continuously positions and velocities of moving objects with varying accuracies, such technologies often rely on proximity analysis [7] and are unable to report velocities or accurate locations.

In particular, an indoor object is detected only when it enters the activation range of a positioning device, e.g., an RFID reader or a Bluetooth base station. Depending on the deployment of devices, such detections occur more or less frequently. As a result, the indoor positioning technologies create much more uncertain tracking data in indoor spaces when compared to outdoor settings.

The differences between the outdoor and indoor settings call for new research on indoor moving objects. This paper covers some of the background and results of such ongoing research. The paper presents a graph-based model for the effective and efficient tracking of indoor moving objects with proximity-based positioning technologies like RFID and Bluetooth. It presents an indexing scheme for on-line indoor moving objects. It also conducts a brief analysis on the inherent uncertainty of indoor moving objects. Finally, it suggests several interesting and important research directions.

The rest of this paper is organized as follows. Section 2 presents a graph-based model for indoor tracking. Section 3 presents foundations for the management of indoor objects, presenting a hashing-based indexing scheme and covering also object location uncertainty. Section 4 concludes and offers research directions.

2 Tracking Indoor Moving Objects

2.1 Symbolic Indoor Positioning

In our setting, each deployed positioning device detects and reports the objects that enter its range at a relatively high sampling rate. For example, in an RFID-based positioning system, an RFID reader can detect objects with passive tags attached. Or in a Bluetooth-based system, a base station can detect objects equipped with a Bluetooth-enabled device. A raw reading of the form $\langle deviceID,\ objectID,\ t\rangle$ states that device $deviceID$ detected object $objectID$ at time t.

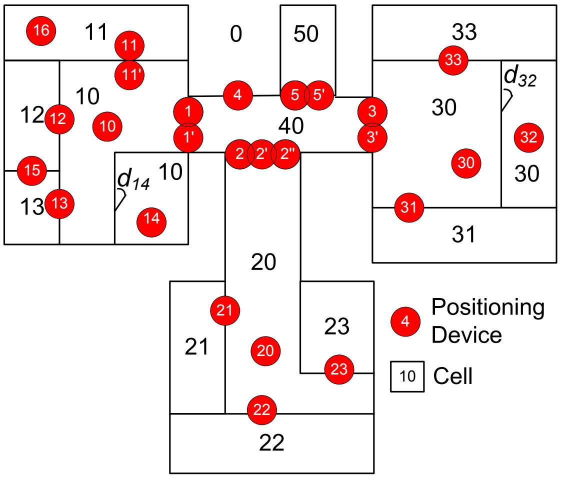


Figure 2: Device Deployment

Figure 2 shows a possible positioning device deployment, where the numbered circles indicate the positioning devices and their activation ranges. For positioning devices with overlapping ranges, we treat the intersections activation ranges of new, virtual positioning devices. Thus, the intersection of $device_1$ and $device_{1'}$ is assigned to a virtual device $device_{1'1}$. An object seen by $device_1$, but not $device_{1'}$, is then in the non-intersecting part of the range of $device_1$.

We also accommodate so-called paired devices (covered in Section 2.2) that detect movement direction, e.g., the entry into or exit from a room.

We apply pre-processing to the raw readings in order to support subsequent on-line and off-line applications. An on-line record is of form $\langle deviceID,\ objectID,\ t,\ flag\rangle$, where $flag = ENTER$ indicates that the object is entering the device's activation range, and $flag = LEAVE$ indicates the object is leaving the range. Note that such records cam be emitted when an object enters or leaves the range of a device with a delay not exceeding the sampling frequency. In contrast, an off-line record is of the form $\langle deviceID,\ objectID,\ t_s, t_e\rangle$, which indicates the presence of the object within the device's activation range during the time interval $[t_s, t_e]$. The details of this pre-processing can be found elsewhere [8].

2.2 Positioning Device Deployment Graph

In a deployment, a subset of the devices, the so-called *partitioning devices*, partition the indoor space into cells in the sense that an object cannot move from one cell to another without being observed. An example is a device deployed by the single door of a room. *Undirected partitioning devices* (*UP*) cannot detect movement directions between cells. In Figure 2, $device_{21}$ cannot tell whether an observed object enters or leaves cell c_{21}. Note that $device_1$, $device_{1'}$, and $device_{1'1}$ are also undirected. In contrast, *directed partitioning devices* (*DP*) consist of entry/exit pairs of sensor that enables the movement direction of an object to be inferred by the reading sequence, e.g., $device_{11}$ and $device_{11'}$ in Figure 2. Finally, *presence devices* (*PR*) simply sense the presences of objects in their ranges, but do not contribute to the space partitioning. Device $device_{10}$ in Figure 2 is a presence device.

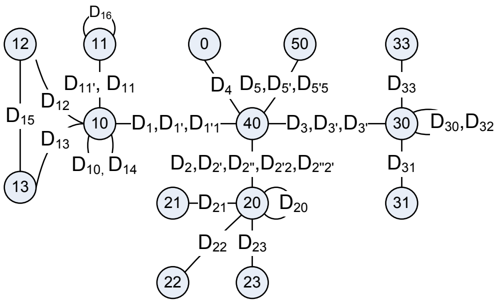


Figure 3: Deployment Graph

To facilitate tracking and querying moving objects, a deployment graph is created based on the topological relationship of the floor plan and the device deployment [8]. Formally, a deployment graph is a labeled graph $G = \langle C, E, \Sigma_{devices}, \ell_E\rangle$, where:

(1) C is a set of vertices corresponding to cells.

(2) E is a set of edges. Each edge is an unordered pair of vertices, indicating that the two cells are connected.

(3) ℓ_E: $E \rightarrow 2^{\Sigma_{devices}}$ assigns a set of devices to an edge. A non-loop edge is labeled by the partitioning device(s) that partition its two cells, and a loop edge captures the presence device(s) in the edge's cell.

Figure 3 shows the deployment graph corresponding to Figure 2; label D_i indicates a positioning device $device_i$.

2.3 Graph Model Based Indoor Tracking

The goal of indoor tracking is to capture the position of an object at any point in time. We propose techniques for both on-line and off-line tracking. By exploiting the indoor floor plan, the deployment graph, and maximum speeds of objects, we try to minimize the possible region(s) an object can be in at a particular time [8]. In doing so, we exploit the deployment graph, which captures the indoor topology that constrains the movements of indoor objects. For example, an object can only move from a graph vertex (a cell) to an adjacent vertex (another cell connected with some partitioning devices).

Given a set of off-line records in the form of $\langle deviceID,\ objectID,\ t_s, t_e \rangle$, off-line tracking of an indoor moving object is conducted in three steps. Step one augments each reading record with corresponding deployment graph elements (vertices or edges) during the time interval $[t_s, t_e]$. Pre-defined mappings between positioning devices and relevant graph vertices (cells) are also used in this step.

Step two identifies cells that an object can possibly be in during its *vacant time intervals*, which are the intervals during which no tracking record exists for the object. Specifically, step one tells where (graph elements) the object is before and after a vacant time interval; its position during the vacant time interval is constrained to the graph elements that connect the before and the after parts. So, by intersecting the graph elements before and after the vacant time interval, we identify the cells the object can be in during the vacant time interval.

Step three makes use of the maximum speeds of the objects and reduces the possible cells obtained in step two to smaller regions. If the object moves at its maximum speed V_{max} from any point inside the activation range of a device and its trajectory is a straight line, its position at time t_x will be bounded by circles centered at all possible start points in the activation range and with radius $V_{max} \cdot \Delta t_1$. Applying this constraint to two consecutive tracking records, the possible region of the object during a vacant time interval can be simplified to an speed-constrained ellipse.

Given a set of records of the form $\langle deviceID,\ objectID,\ t,\ flag \rangle$, on-line tracking treats the cases where *flag* is either $ENTER$ or $LEAVE$ differently. Details can be found elsewhere [8].

3 Management of Indoor Moving Objects with Inherent Uncertainty

3.1 Indexing of Indoor Moving Objects

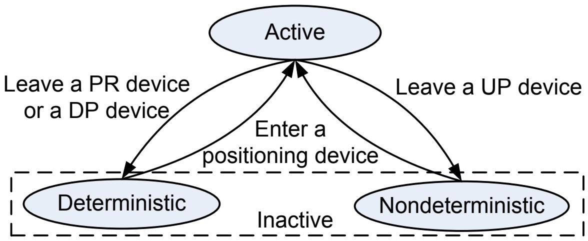


Figure 4: Object State Transition Diagram

An object may be active or inactive. An *active object* is currently seen by at least one positioning device, while an *inactive object* is currently not seen by any positioning device. The latter are further divided into *deterministic objects* that must be in one specific cell and *nondeterministic objects* that may be in more than one cell. An object can change state according to the diagram shown in Figure 4.

The consequent partitioning of objects can be exploited in a hashing-based object-location indexing technique. Let O_{indoor} be the set of all the moving objects in the indoor space of interest. A *Device Hash Table (DHT)* maps each positioning device, identified by $deviceID$, to the set of active objects in its range:

$$DHT[deviceID] = O_A; \quad deviceID \in \Sigma_{devices}, O_A \subseteq O_{indoor}$$

Next, a *Cell Deterministic Hash Table (CDHT)* maps each cell, identified by $cellID$, to the set of deterministic objects in it:

$$CDHT[cellID] = O_D; \quad cellID \in C, O_D \subseteq O_{indoor}$$

Similarly, a *Cell Nondeterministic Hash Table (CNHT)* maps a cell to the set of nondeterministic objects in it:

$$CNHT[cellID] = O_N; \quad cellID \in C, O_N \subseteq O_{indoor}$$

Finally, an *Object Hash Table (OHT)* captures the states of all objects:

$$OHT[objectID] = (STATE,\ t,\ IDSet); objectID \in O_{indoor}$$

Here $STATE$ denotes the object's current state and t is the start time of the state. If the object's state is active, $IDSet$ is a singleton set of a device identifier. If the state is deterministic, $IDSet$ is a singleton set of a cell identifier. If the state is nondeterministic, $IDSet$ is a set of cell identifiers.

The update of these hash tables and their use in query processing are covered elsewhere [14]. Also, it is possible to extend the R-tree to index large volumes of historical indoor tracking data [9].

3.2 Uncertainty Analysis for Indoor Moving Objects

As for outdoor moving objects [5], the uncertainty region of an indoor object o at time t, denoted by $UR(o, t)$, is a region such that o must be in this region at time t. The uncertainty region of an active object is the activation range of the corresponding device, while the uncertainty region of an inactive object is the cell or cells that the object can belong to.

If the object's maximum speed V_{max} is given, its uncertainty region can be captured at a finer granularity. The uncertainty region of a deterministic object is refined as the intersection between the object's cell and its *maximum-speed constrained circle*. For a nondeterministic object, the region is the union of the intersections between each cell and the circle.

Let the last $LEAVE$ observation of object o be from device dev at time t and let the duration from t to the current time be $\Delta t = t_{now} - t$. The longest possible distance o can move away from the boundary of dev's activation range is $o.V_{max} \cdot \Delta t$. Formally, the maximum-speed constrained circle $C_{MSC}(o, dev, t)$ of o is defined as the circle centered at dev's deployment location and with radius $o.V_{max} \cdot \Delta t$ plus the radius of dev's activation range. We also exclude the activation range of dev from the circle.

Consider Figure 5 and assume that object o left $device_{16}$ at time t. Its maximum-speed constrained circle $C_{MSC}(o, device_{16}, t)$ is then indicated by R_1 in the figure. Since $device_{16}$ is a presence device, after leaving $device_{16}$ the inactive object o must be in the cell c_{11} (according to $G.\ell_E^{-1}(device_{16})$). Due to the two constraints, object o's uncertainty region is the intersection of cell c_{11} and circle R_1, i.e., the shaded region in the top-left part of Figure 5.

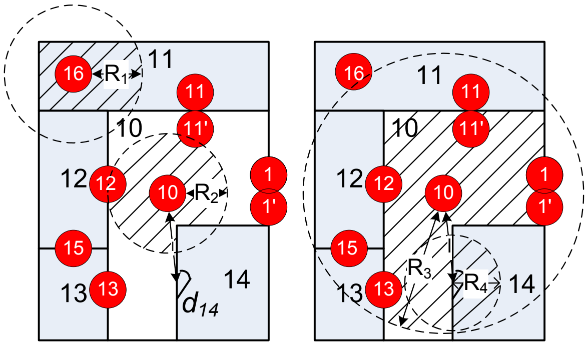


Figure 5: Uncertainty Regions

If the cell where the deterministic object resides has more than one room, e.g., cell c_{10} contains rooms 10 and 14, the determination of the uncertainty region is more complicated. Suppose object o left $device_{10}$ at time t. According to $G.\ell_E^{-1}(device_{10})$, o should be in cell c_{10} after leaving $device_{10}$. From predefined mappings that capture the deployments of devices [15], it follows that $device_{10}$ resides in room 10 and that the distance from $device_{10}$ to door d_{14} is l. If the maximum speed constraint guarantees that o cannot have gone through door d_{14}, i.e., $o.V_{max} \cdot (t_{now} - t) < l$, the object o must remain in room 10. Thus, the uncertainty region is the intersection between room 10 and $C_{MSC}(o, device_{10}, t)$, which is indicated by R_2 to the left in Figure 5.

On the other hand, referring to the right part of Figure 5, if $o.V_{max} \cdot (t_{now} - t) \geq l$, object o may have entered room 14. Its uncertainty region therefore contains two parts: the intersection between room 10 and $C_{MSC}(o, device_{10}, t)$ (indicated by R_3); and the intersection between room 14 and the circle with door d_{14} as the center and $R_4 = o.V_{max} \cdot (t_{now} - t - l/o.V_{max})$ as the radius.

The uncertainty region of an active object can also be refined in the similar way [14, 15].

The online indexing scheme and the uncertainty analysis have been used for processing queries on indoor moving objects, e.g., indoor range monitoring [14] and indoor k nearest neighbor queries [15].

4 Conclusion and Future Work

Indoor spaces differ substantially from outdoor spaces and are not modeled well by Euclidean spaces or spatial networks. Further, indoor positioning may be accomplished by presence-sensing technologies rather than the GPS-like positioning that is often assumed in research targeting outdoor settings. Due to these and other factors, "indoor" offers new research challenges.

This paper offers a glimpse of selected aspects of the foundations for ongoing research on data management for indoor moving objects. It touches upon graph model based indoor tracking, indoor moving-object indexing, and the capture of the uncertainty of indoor moving objects.

There are many research opportunities in data management for indoor spaces. Here, we mention but a few.

- It is of interest to integrate different types of positioning technologies in order to improve tracking accuracy. For example, we may combine proximity analysis based on RFID and Bluetooth with fingerprinting-based technologies like Wi-Fi [2]. This may yield an augmented graph model [8] of indoor space.
- In addition to relying on symbolic locations for co-location queries, it is of interest to accommodate distances in indoor models. This may enable distance-aware queries. For example, it becomes possible to monitor closest pairs of indoor moving objects. As another example, given a distance value e, an e-distance join returns those pairs of objects whose distance is smaller than e. Distance-aware queries may have security and social-network applications.
- Given large volumes of real tracking data, it is interesting to mine patterns or association rules. This may enable on-line prediction of aggregate and individual movements, which in turn may improve on-line tracking accuracy. It may also serve to improve query processing efficiency.
- While initial research has assumed that objects move independently, it is of relevance to consider more advanced models of object movement. For example, it is relevant to conduct probabilistic analyses that assume Gaussian distributions.
- It is worth developing benchmarks for indoor moving object data management that enable the comparison of competing techniques. Relevant aspects include, but are not limited to, indoor space and positioning device configuration, object movement workload generation, and query workload generation.

Acknowledgments

This research was partially supported by the Indoor Spatial Awareness project of the Korean Land Spatialization Group and BK21 program. C. S. Jensen is an adjunct professor at University of Agder, Norway.

References

[1] P. K. Agarwal, L. Arge, and J. Erickson. Indexing Moving Points. In *Proc. PODS*, pp. 175–186, 2000.

[2] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proc. INFOCOM*, pp. 775–784, 2000.

[3] C. Becker and F. Dürr. On Location Models for Ubiquitous Computing. *Personal Ubiquitous Computing*, 9(1):20–31, 2005.

[4] S. Feldmann, K. Kyamakya, A. Zapater, and Z. Lue. An Indoor Bluetooth-Based Positioning System: Concept, Implementation and Experimental Evaluation. In *Proc. ICWN*, pp. 109–113, 2003.

[5] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *IEEE TKDE*, 16(9):1112–1127, 2004.

[6] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Quering Moving Objects. *ACM TODS*, 25(1):1–42, 2000.

[7] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, 2001.

[8] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *Proc. MDM*, pp. 122–131, 2009.

[9] C. S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *Proc. SSTD*, pp. 208–227, 2009.

[10] G. Kollios, D. Gunopulos, and V. J. Tsotras. On Indexing Mobile Objects. In *Proc. PODS*, pp. 261–272, 1999.

[11] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. In *Proc. SIGMOD*, pp. 331–342, 2000.

[12] M. Pelanis, S. Šaltenis, and C. S. Jensen. Indexing the Past, Present, and Anticipated Future Positions of Moving Objects. *ACM TODS*, 31(1):255–298, 2006.

[13] R. Want. RFID Explained: A Primer on Radio Frequency Identification Technologies. *Synthesis Lectures on Mobile and Pervasive Computing*, 1(1):1–94, 2006.

[14] B. Yang, H. Lu, and C. S. Jensen. Scalable continuous range monitoring of moving objects in symbolic indoor space. In *Proc. CIKM*, pp. 671–680, 2009.

[15] B. Yang, H. Lu, and C. S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *Proc. EDBT*, pp. 335–346, 2010.

Spatio-temporal Databases in Urban Transportation*

Ouri Wolfson and Bo Xu
University of Illinois, Chicago, IL 60607
{wolfson,boxu}@cs.uic.edu

Abstract

In this paper we describe applications, research issues, and approaches related to Intelligent Transportation Systems (ITS). More specifically, we focus on spatio-temporal databases in urban transportation. We address the issues of trip planning and navigation, abstraction of concepts from spatio-temporal sensor data, mobile peer-to-peer data management, and social networks and crowd-sourcing. These issues are strongly related to ITS efforts currently undertaken throughout the world, particularly the IntelliDrive initiative of the US Department of Transportation.

1 Introduction

The impact of Computer Science (CS) and Information Technology (IT) on transportation systems is not as dramatic as the one on finance or business in general. But in the last few years we have witnessed significant penetration of IT in surface transportation. Navigation systems with real-time traffic information, color coded traffic maps, and real-time information about public transportation vehicles (e.g. Nextbus and Chicago Transit Authority's bus-tracker) are some examples of the improvements in urban transportation brought about by IT. Rapid advances in mobile and ubiquitous computing, sensor networks, and Geographic Information Systems are creating opportunities to revolutionize urban transportation. Indeed, the purpose of the IntelliDrive initiative of the U.S. Department of Transportation is "advancing connectivity among vehicles and roadway infrastructure in order to significantly improve the safety and mobility of the U.S. transportation system" [18]. Additional compatible goals are reduction of environmental impact and energy consumption.

In the envisioned environment, billions of sensors embedded in the infrastructure, in portable devices, and in vehicles will generate vast amounts of data whose interpretation could be exploited to spur the creation of innovative transportation services and policies. Advances in social networking and data mining research are increasingly creating new sophisticated mechanisms which can foster seamless information integration among travelers, provide alternatives, and support sustainable economic and social policies.

In this paper we discuss data management issues in the envisioned environment. Specifically, we discuss the applications, research issues, and approaches in the following research areas.

- **Route Planning (Routing), Navigation, and Tracking.** This area studies methods of routing, navigation, and tracking in transportation networks (the spatio-temporal database) that may involve multiple modes

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Research supported by NSF DGE-0549489, IIS-0957394, and IIS-0847680.

such as train, bus, private car, and bicycle. Optimization criteria are traditional such as time, distance, and cost, but may also involve environmental and energy aspects. For example, the requested route should minimize exposure to, and/or generation of, pollution. Data models, query languages, and processing algorithms are key research issues. The maintenance of dynamic, up-to-date, and reliable travel time and incident information is another important research issue.

- **Abstraction of concepts from spatio-temporal sensor data.** This area studies mining techniques for analyzing vast amounts of data related to moving objects. For example, map matching (i.e., matching the GPS traces of a traveler to the road network) extracts route information. A higher level abstraction would be extraction of semantic location and activity knowledge from GPS traces, possibly augmented with other information such as visited-web-sites lists. This analysis may significantly improve the accuracy and efficiency of household activity surveying, which is an important data source for transportation planning.

- **Mobile peer-to-peer data management.** This area studies the problem of querying and dissemination of spatio-temporal traveler information via short-range wireless communication among vehicles, pedestrians, and road side facilities. The spatio-temporal traveler information includes, for example, accidents, transit vehicle on-time performance, parking availability, and ride share opportunities. The key research issue is query processing in a distributed and mobile ad hoc environment, without data-directory services. Bandwidth, energy, and memory constraints complicate the problem.

- **Social networks and crowd-sourcing.** This area studies techniques based on social networks to crowd-source information that is valuable to travelers, traffic administrators, and long-range transportation planners. For example, traffic congestion, parking slot availability, ride share opportunities, and cooperative driving [29] information will be obtained and disseminated via (possibly ad hoc) social networks. A major research issue is providing incentives for participation.

In sections 2 to 5, we discuss each of the above research areas separately.

2 Route Planning (Routing), Navigation, and Tracking

2.1 Applications

For trip planning, travelers often ask queries that pertain to routes involving various constraints, optimization criteria together with uncertainty clauses (see e.g., [21]). For example, the queries can be: ”Find a route that will get me home by my designated time with at least 90% certainty”; or ”Using public transportation, find a route that lets me stop at a grocery store for 30 minutes and reach home by 7:00pm”. A special application that has been studied extensively is evacuation route planning: given a transportation network, locations of a vulnerable population, and a set of destinations, identify exit routes to minimize the time to evacuate the vulnerable population [12]. While the planned trip is being executed, it is often useful to track the traveler, an activity that consumes resources such as bandwidth, energy, and processing power.

2.2 Research Issues and Approaches

Relevant research issues include: 1. data models and appropriate query languages to represent the routing problem; 2. performance of the routing and tracking algorithms (see e.g. [25]), and their incorporation into the query processing engine; 3. efficient and effective collection and distribution of travel-time information; 4. travelers are usually interested in travel time of a road link when they get to it, so it is important to *predict* travel-time based on past travel times, and based on travel-times on near-by road links (see e.g., [13]); 5. scalability and performance of transportation network micro-simulators such as CORSIM, VISSIM, and TRANSSIM with increase of network size, traffic intensity, and fidelity of vehicular behavior [16]. In the rest of this subsection we will elaborate on existing approaches to some of the above problems.

2.2.1 Transportation network data models, query languages, and processing

One technical difficulty with multi-modal public-transportation trip planning is the heterogeneous nature of the data and the lack of a coherent data model that can be used effectively. In [4] we proposed a method to integrate the key aspects of spatio-temporal, moving objects, and graph-based databases to facilitate trip planning in multi-modal urban transportation networks. Our solution is based on a graph model of the network. This graph model contains not only topology of the network but also various other information such as real-time traffic information including predicted travel times, dynamic schedule information, available facilities, etc. A trip corresponds to a path from an origin to a destination in this graph.

We defined the language for specifying queries in our graph model. The language focuses on querying trips subject to various constraints – including uncertainty of travel times on the links. The query structure extends the basic syntax of SQL with additional clauses. For example, the following query retrieves trips/paths from work to home where only the pedestrian and bus modes are allowed. The CERTAINTY and WHERE clauses specify that the trip must end by 5:00 pm with probability greater than 0.8. Among all such paths the shortest path (by length) is selected.

*SELECT * FROM ALL_TRIPS(work, home) AS t WITH MODES pedestrian, bus WITH CERTAINTY > .8 WHERE FINISHES(t) <= 5:00pm MINIMIZE LENGTH(t)*

The reader is referred to [4] for the detailed syntax and semantics of the query language. That work also presents algorithms for processing a restricted class of queries specified in this language, and proposes a plug-and-play approach in which existing and new algorithms developed in the Operations Research field can be incorporated. For a prototype implementation see [3].

2.2.2 Maintenance of link travel-times

Accurate information about travel-times on road-links of the transportation network is critical for trip planning.
Travel-times Collection. Point detection by inductive loop detectors, ultrasonic detectors, remote traffic microwave sensors, video cameras, etc. is a common travel-time collection method. These devices are typically deployed on selected highways, but not on urban streets for both economical and technical causes. A promising alternative to point detection is floating car data (FCD), or probe car data, which uses probe vehicles to collect travel-time information. FCD can cover both highways and arterial roads without dedicated sensor deployment. A similar method is to use mobile phones to collect travel-time information as demonstrated by the Mobile Millennium project (traffic.berkeley.edu). For this method a technical challenge is to identify and eliminate the data generated by pedestrians. In the rest of this subsection we focus on FCD.

In terms of system architecture, there are two ways to collect floating car data. One way uses a client-server architecture, where vehicles (clients) send the travel-time that they experience to the server. This informs the server of the real time traffic condition of each road segment. In turn, the server broadcasts updated travel-times to the vehicles. In [2] we proposed methods that reduce communication by using randomization. The other way is based on a mobile peer-to-peer (P2P) architecture, where vehicles send travel-times of road segments to each other via vehicle-to-vehicle communication, without server facilities (see TrafficView [8] and SOTIS [24]).

Travel-times Aggregation. Collected travel-times may be aggregated in different ways for different purposes. In mobile P2P FCD, travel-times may be aggregated to reduce the communication volume [8]. In this case, the level of aggregation varies such that the travel-time information is less accurate about regions that are farther away from the vehicle at which the aggregation occurs. In client-server FCD, travel-times may be aggregated to generate a travel-time profile of a link, namely how the travel-time of the link varies depending on the time of a day, the day of a week, etc. Trip planning based on time-dependent travel-times enables more efficient navigation. The authors in [17] proposed a data warehouse for building travel-time profiles. The basic fact table contains the collected travel-time data. Travel-times are aggregated along the spatial and temporal dimensions to create a data warehouse enabling efficient processing of queries such as ”What is the average travel-time of the links in downtown Chicago on Mondays from 9:00-9:15?”.

3 Abstraction of Concepts from Spatio-temporal Sensor Data

3.1 Applications

The transportation environment consists of various mobile sensors such as on-board GPS receivers, sensors mounted on public transportation vehicles (to monitor, e.g., traffic, air quality, pollution, and toxic gases) or pedestrian cell phones. These sensors continuously generate spatio-temporal data and enable applications such as moving objects tracking and environmental monitoring [10]. The abstraction of concepts from spatio-temporal data can derive, for example, the context of a traveler which indicates where she is and what she is doing. The context information may be used to reduce user burden when dealing with the restricted user interfaces of mobile devices. For example, if the traveler’s smartphone finds that its owner is at an electronic store, then the smartphone may automatically retrieve appropriate coupons. The context information may also be used to complement the traditional paper based household travel survey by automated activity detection. Another application of concepts abstraction is to identify energy efficient driving patterns, and optimal routing patterns for minimization of pollution and energy consumption.

3.2 Research Issues and Approaches

Relevant research issues include: 1. data mining to extract semantic location and activity knowledge from sensor traces; 2. efficient monitoring of spatio-temporal streams in terms of communication, storage, and query processing; 3. management of uncertainty which is inherent to continuously changing variables such as traveler locations; 4. fusion of information from multiple sensors generating heterogeneous data. In the rest of this subsection we will elaborate on existing approaches to some of the above problems.

3.2.1 Map matching

For the purpose of abstraction, usually GPS reported locations need to be mapped to the road network. This procedure is called *map matching*. A straightforward way to do map matching is to snap each GPS point to the closest road segment. However, this method often produces incorrect results [28, 11]. A better method is to snap subsequences of GPS points in 3D space-time. For example, in [28] we proposed an algorithm based on a weighted graph representation of the road network in which the weight of each road link represents the distance of the link to the trajectory. The matched route in the road network is then found by computing the shortest-path in the weighted graph.

3.2.2 Extracting semantic locations from GPS traces

A semantic location carries context information about the physical (x,y) location, e.g., ”office at 851 S. Morgan St”, or ”grocery store at 1340 S. Canal St”. In [14] we proposed a method to extract the semantic locations from traces of GPS points augmented with behavioral information. The principle is as follows. First, we extract some 2D positions where the user stays for a while. Then, we use reverse geocoding (i.e., translating a physical location to a street address) to obtain the street address candidates for each stay. Finally, correlating with other available sources (such as semantic locations history, map, yellow pages, address book, calendar, phone-calls trace, visited web pages trace), we determine the semantic location of a stay.

3.2.3 Automated survey data reduction

In traditional travel surveys, respondents have difficulties giving accurate data due to cumbersome data entry requirements and recall limitations. An effective approach to reducing respondents' burden and increasing data accuracy is to automatically detect activities from GPS traces, and prompt them for user verification. This approach is called *prompted recall surveying* [1]. Activities may be detected using sequential association mining techniques (see e.g., [23]). Specifically, a respondent's activity schedule over a time period (e.g., each day) is modeled as a sequence. Association rules are learned from historical sequences, e.g., ”with 80% probability the traveler goes to a restaurant after shopping on Friday”. Using these, and given some observations that have been filled up in a new sequence, the successive activities can be predicted.

3.2.4 Uncertainty in moving objects databases

A moving objects database often tracks the location of vehicles and travelers. The objective of uncertainty management is to assist the user in accounting for spatio-temporal uncertainty, and in expressing imprecise queries/triggers. Uncertainty management is often the first step in abstracting sensor-data.

For example of an uncertain query, a trucking company may ask: ”Retrieve the current location of the delivery trucks that will possibly be inside a region R, sometime between 3:00PM and 3:15PM”. In [22] we proposed operators for uncertain spatio-temporal range queries, and their processing algorithms. Computational geometry played an important role in these algorithms. In [15], the uncertainty is modeled quantitatively by probabilistic values. This enables answering the queries such as: ”What is the probability that a given object will be inside a given region R sometime between t_1 and t_2?”

3.2.5 Compression/abstraction of spatio-temporal data

A key observation that lies at the foundation of spatio-temporal data compression is that a GPS point (x, y, t) can be eliminated if (x, y, t) can be approximated with a reasonable accuracy by interpolating the adjacent (i.e., before and after) GPS points. In [6] we formalized this intuition by employing a mechanism based on line simplification, which has been studied in computational geometry, cartography and computer graphics. Basically, line simplification approximates a polygonal line by another that is ”sufficiently close”, and has less straight-line segments (or points). The attractiveness of line simplification (compared to other lossy data compression techniques such as wavelets) stems from the fact that the approximation carries a given error bound. However, we discovered that although the approximation error is bounded, the error of the answers to queries may not be bounded. Whether or not it is bounded, depends on the combination of the distance function (or distance for short) used in the approximation, and the spatio-temporal query type. Furthermore, we considered an aging mechanism by which a trajectory is represented by increasingly coarser abstractions as time progresses.

4 Mobile P2P Data Management

4.1 Applications

A mobile peer-to-peer (MP2P) database resides on a set of mobile peers that communicate with each other via short-range wireless protocols, such as IEEE 802.11 and Bluetooth. The peers communicate reports (data) and queries to neighbors directly, and the reports and queries propagate by transitive multi-hop transmissions. The mobile P2P database provides various types of real-time traveler information, including safety alerts (e.g., a car in front with a malfunctioning brake light), traffic conditions (possibly represented by multimedia clips), accidents, transit-vehicle on-time performance, parking availability, ride share opportunities, audio and video clips of traffic conditions, etc. These applications are more amenable to mobile P2P dissemination than client/server because of real-time safety requirements. Also, since the network id's of the destination nodes are often unknown, the limited transmission range of the network is used instead.

Observe that the MP2P database stores spatio-temporal information, and that the query/trigger processing algorithm does not guarantee to produce all the answers. However, as more data and queries are being disseminated, the throughput and response time improve.

4.2 Research Issues and Approaches

Relevant research issues include: 1. query processing in a distributed and mobile ad hoc environment without data-directory services; 2. query processing with storage, bandwidth, and energy constraints, particularly in managing multimedia data; 3. integration of mobile P2P communication and infrastructure communication; 4. analysis of how reports are propagated in space and time.

4.2.1 Query processing with resource constraints

Given energy and bandwidth constraints, prioritization of reports is critical, so that maximum useful traveler information is disseminated. A useful report is one that has an impact on the decision making process of the receiver. For example, in a travel-time dissemination application, a report is useful if it changes the shortest-route of a vehicle. In a parking space discovery application, a report is useful if it leads to a successful occupation of the reported parking space. The usefulness of a report can be estimated by a mobile node based on its characteristics such as the age, distance, and application. In [19], we demonstrated the feasibility and advantages of using online machine learning algorithms to determine the relevance (i.e. probability of being useful) of a report. The general idea is to use the received reports as an input to a supervised machine learning process. The interesting aspect is that the supervision can be carried out automatically, without user intervention. For example, an algorithm running on a vehicle can make judgments regarding the usefulness of a report. Over time, each vehicle learns a model that can estimate the relevance of a report, and the model can then be used as a ranking function.

4.2.2 Querying blobs in mobile P2P databases

When data reports include binary large objects (blobs) such as video/voice clips, query processing in MP2P databases becomes even more challenging. However, for the purpose of matchmaking, namely finding the reports that satisfy a query, only the metadata descriptions of the reports are needed. For example, the metadata of a multimedia clip may simply include the time and location at which the clip was produced. In such an environment, the design choices for query processing can be made along multiple dimensions. One dimension considers that the mobile P2P communication may use purely short-range or it may use both short-range and cellular communication. Another dimension considers that, due to size-differences, the metadata and blob subreports of a given report may be disseminated independently, and by different means. In [26] we analyzed the blob query-processing strategies along these dimensions.

4.2.3 Propagation Analysis

The fundamental question that the propagation analysis answers is: given a mobile P2P database and its properties in terms of peer density, mobility, dissemination method, and storage/bandwidth/energy constraints, what is the probability that a peer at location (x, y) has received a report R that was generated at location (x_0, y_0) t time units ago. This probability is useful for multiple purposes. For example, it helps ranking because if this probability is high then the rank of R should be lowered. It also enables the comparison of various dissemination methods without having to conduct simulations.

Epidemiology renders some methodology to propagation analysis since mobile P2P dissemination is to some extent similar to the spreading of epidemic diseases. However, mobile P2P dissemination is much more complex because many diseases (reports) are spread at the same time, and the spreading of one interferes with the other (competing for bandwidth and storage). In [20] we conducted an analysis with bandwidth and storage constraints taken into account. The analysis is based purely on the age of a report, and extension to both age and distance remains an open problem.

5 Social Networks

Social networks can be used to crowd-source information and to disseminate a variety of valuable spatio-temporal information to the traveler and the traffic administrator. For example, traffic congestion, parking slot availability, ride sharing opportunities, and cooperative driving information can be obtained and disseminated via (possibly ad hoc) social networks.

Relevant research issues include: 1. providing incentives for participation; 2. quality of information supplied by non-liable crowds; 3. mitigating privacy loss when crowds have to disclose their locations [7].

Now consider incentives. In general, there are three approaches to incentivizing users to participate in crowd-sourcing. The first approach, called *pricing incentive*, remunerates participating users using a virtual currency scheme (see e.g., [27]). The decisions to make are who pays, who charges, and how much is paid or charged in various transactions. When crowd-sourcing is implemented in a mobile P2P fashion, an open issue is the atomicity problem, which occurs when a trading transaction between two users needs to be settled even if the users are disconnected before the transaction completes [27]. The second approach, called *soft-incentive* or *non-pricing incentive*, denies services to non-participating users using a reputation scheme (see e.g., [5]. The third approach used in ad hoc networks assumes, based on game theory, that rational users are incentivized to participate (see e.g., [9]). It is not clear whether this approach is feasible in a transportation environment, or can be made feasible without resorting to some form of reputation management.

References

[1] J. Auld, Chad A. Williams, Abolfazl Mohammadian, and Peter C. Nelson. An Automated GPS-Based Prompted Recall Survey With Learning Algorithms. *Transportation Letters*, 1(1), Jan. 2009, pp. 59-79.

[2] D. Ayala, J. Lin, O. Wolfson, N. Rishe, and M. Tanizaki. Communication Reduction for Floating Car Data-based Traffic Information Systems. *GeoProcessing*, 2010.

[3] J. Biagioni, A. Agresta, T. Gerlich, and J. Eriksson. Transitgenie: A real-time, context-aware transit navigator (demo abstract). In *SenSys*, pages 329-330, ACM, 2009.

[4] J. Booth, A. P. Sistla, O. Wolfson, and I. Cruz. A Data Model and Query Language for Urban Transport Systems. *12th Int. Conf. on Extended Database Technology*, Mar. 2009, pages 994-1005.

[5] S. Buchegger and J. Boudec. A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. *Proceedings of P2PEcon*, Harvard University, Cambridge MA, U.S.A., June 2004.

[6] H. Cao, O. Wolfson, G. Trajcevski. Spatio-Temporal Data Reduction with Deterministic Error Bounds. *The VLBD Journal*, Vol.15(3), Sept. 2006, pp. 211-228.

[7] C. Cottrill. Approaches to Privacy Preservation in Intelligent Transportation Systems and Vehicle-Infrastructure Integration Initiative. In *Transportation Research Record*, 2129:9-15, 2010.

[8] S. Dashtinezhad, T. Nadeem, C. Liao, and L. Iftode. Trafficview: A scalable traffic monitoring system. *IEEE Int. Conf. on Mobile Data Management*, January 2004.

[9] M. Felegyhazi, J. Hubaux, and L. Buttyan. Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(5), 2006, pp. 463-476.

[10] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *Proc. ACM SenSys*, Nov. 2006.

[11] C. Jensen and N. Tradiauskas. Map Matching. *Encyclopedia of Database Systems*, 2009, pp. 1692-1696.

[12] S. Kim, B. George, S. Shekhar. Evacuation route planning: scalable heuristics. *ACMGIS*, 2007.

[13] H. Lin, R. Zito, and M. Taylor. A Review of Travel-time Prediction in Transport and Logistics. Eastern Asia Society for Transportation Studies, 2005.

[14] J. Liu, and O. Wolfson, H. Yin. Extracting Semantic Location from Outdoor Positioning Systems. *Int. Workshop on Managing Context Information and Semantics in Mobile Environments (MCISME)*, 2006.

[15] H. Mokhtar, J. Su. Universal Trajectory Queries for Moving Object Databases. *MDM* 2004.

[16] K. Perumalla. A systems approach to scalable transportation network modeling. *Proceedings of the 38th conference on Winter simulation*, 2006.

[17] D. Pfoser, N. Tryfona, and A. Voisard. Dynamic Travel Time Maps – Enabling Efficient Navigation. *Proc. 18th SSDBM*, 2006, pp. 369-378.

[18] Research and Innovative Technology Administration. *ITS Strategic Research Plan*, 2010-2014. Executive Summary. http://www.its.dot.gov/strat_plan/pdf/ITSStrategicResearch_Jan2010.pdf

[19] P. Szczurek, B. Xu, J. Lin, O. Wolfson. Machine Learning Approach to Report Prioritization with an Application to Travel Time Dissemination. *IWCTS*, 2009.

[20] P. Sistla, O. Wolfson, B. Xu. Opportunistic Data Dissemination in Mobile Peer-to-Peer Networks. *Proceedings of the 9th International Symposium on Spatial and Temporal Databases*, 2005, pp. 346-363.

[21] M. Schubert, M. Renz, and H. Kriegel. Route Skyline Queries: A Multi-Preference Path Planning Approach. *ICDE*, 2010.

[22] G. Trajcevski, O. Wolfson, K. Hinrichs, S. Chamberlain. Managing Uncertainty in Moving Objects Databases. *ACM Transactions on Database Systems (TODS)*, 29(3), Sept. 2004, pp. 463-507.

[23] C. A. Williams, P. C. Nelson, and A. Mohammadian. Attribute Constrained Rules For Partially Labeled Sequence Completion. *Advances in Data Mining - Applications and Theoretical Aspects*, 2009.

[24] L. Wischoff, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. SOTIS - a self-organizing traffic information system. *Vehicular Technology Conference*, 2003.

[25] O. Wolfson, H. Yin. Accuracy and Resource Consumption in Tracking Moving Objects. *SSTD*, 2003.

[26] O. Wolfson, B. Xu, H. Cho. Multimedia Traffic Information in Vehicular Networks. *ACMGIS*, 2009.

[27] B. Xu, O. Wolfson, and N. Rishe. Benefit and Pricing of Spatio-temporal Information in Mobile Peer-to-Peer Networks. *HICSS-39*, 2006.

[28] H. Yin, O. Wolfson. A Weight-Based Map Matching Method in Moving Objects Databases. *The 16th International Conference on Scientific and Statistical Database Management (SSDBM)*, 2004.

[29] B. van Arem, C. van Driel, and R. Visser. The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics. *IEEE Transactions on Intelligent Transportation Systems*, (7)4:429-436, 2006.

Evacuation Planning: A Spatial Network Database Approach

Xun Zhou[1], Betsy George[2], Sangho Kim[3], Jeffrey M. R. Wolff[1], Qingsong Lu[1], Shashi Shekhar[1]
[1]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA
{xun, jwolff, lqingson, shekhar}@cs.umn.edu
[2]Oracle, Nashua, NH, USA
betsy.george@oracle.com
[3]Geodatabase Team, ESRI, Redlans, CA, USA
skim@esri.com

Abstract

Efficient tools are needed to identify routes and schedules to evacuate affected populations to safety in face of natural disasters or terrorist attacks. Challenges arise due to violation of key assumptions (e.g. stationary ranking of alternative routes, Wardrop equilibrium) behind popular shortest path algorithms (e.g. Dijkstra's, A) and microscopic traffic simulators (e.g. DYNASMART). Time-expanded graphs (TEG) based mathematical programming paradigm does not scale up to large urban scenarios due to excessive duplication of transportation network across time-points. We present a new approach, namely Capacity Constrained Route Planner (CCRP), advancing the idea of Time-Aggregated Graph (TAG) to provide Earliest-Arrival-Time given any Start-Time. Laboratory experiments and field use in Twin-cities for Homeland Security scenarios show that CCRP is more efficient than previous methods.*

1 Introduction

Evacuation planning is a crucial task for managing public safety. Whether natural (flood, hurricanes, etc) or man-made (release of chemical or toxic substances, etc) disasters require that emergency personnel be able to move affected populations to safety in as short a time as possible. Despite the increased threat of disasters posed by global climate change and the rise of terrorism, however, current evacuation planning tools have serious limitations. Evacuation conducted during Hurricane Katrina and Rita in 2005 were a stark reminder of how much it can go wrong despite intensive emergency population. For example, Figure 1 shows the miles of backed-up traffic that occurred as Houston residents followed orders to flee the path of Hurricane Rita. Therefore, efficient tools are needed to produce plans that identify optimal evacuation routes and schedules, given a transportation network, node/edge capacity constraints, source/destination nodes and evacuee population. Current methods of evacuation planning can be

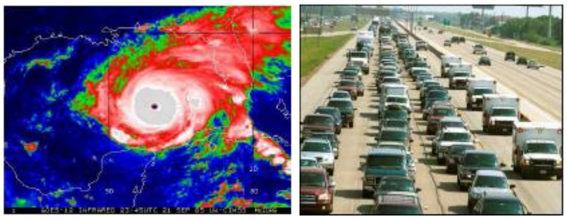


Figure 1: Hurricane Rita and Evacuation Traffic. Source: National Weather Services and FEMA

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

divided into two categories, namely traffic assignment-simulation and route-schedule planning. Micro-scopic traffic assignment-simulation [2] often requires a long time to complete for a large transportation network, and mostly employ the Wardrop equilibrium model [3] which will be violated in emergencies. The route-schedule planning approaches use network flow and routing algorithms to produce routes and schedules, and obtain the optimal solution by using linear programming (LP) based methods. An extensive literature review of these methods was proposed by Hamacher and Tjandra [5]. However, this kind of approach significantly increases the problem size and does not scale well to large transportation networks. It also requires the user to provide an upper bound T of the evacuation time which is almost impossible to precisely estimate.

We propose the Capacity Constrained Route Planner (CCRP) to efficiently solve the evacuation planning problem in large scale of transportation networks, with the capacity of the network taken into consideration. This paper gives an overview of the evacuation planning problem as well as the CCRP system, and shows the utility of CCRP in real scenarios.

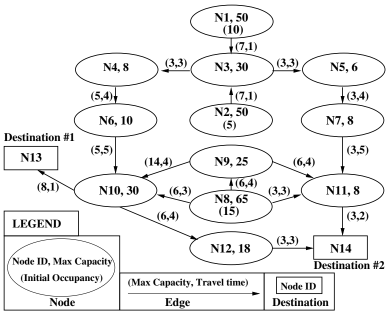
2 Problem Statement

Transportation networks are usually modeled as graphs. Nodes represent the intersections of roads, and edges represent the road segments between them. Cost of edges represent their travel time. The capacity attribute associated with each edge represents the maximum flow rate (flow per unit of time), and the capacity of each node indicates the maximum number of evacuees that can be held at this node. Source nodes are the initial starting points of evacuees, and destination nodes are the safe places that they are supposed to arrive finally. The evacuation planning problem is formulated as follows:

Given: A transportation network with non-negative integer capacity constraints on nodes and edges, non-negative integer travel time on edges, the total number of evacuees and their initial locations, and locations of evacuation destinations.
Output: An evacuation plan consisting of a set of origin-destination routes and a scheduling of evacuees on each route. The scheduling of evacuees on each route should observe the capacity constraints of the nodes and edges on this route.
Objective: (1) Minimize the evacuation egress time, which is the time elapsed from the start of the evacuation until the last evacuee reaches the evacuation destination. (2) Minimize the computational cost of producing the evacuation plan.
Constraint: (1) Edge travel time preserves FIFO (First-In First-Out) property. (2) Edge travel time reflects delays at intersections. (3) Limited amount of computer memory.

The following example illustrates the input and output of the problem. In the evacuation network shown in Figure 2 (a), each node is shown by an ellipsis. Each node has two attributes: a maximum node capacity and an initial node occupancy (in the number of evacuees). In Figure 3, each edge, shown as an arrow, represents a link between two nodes. Each edge also has two attributes: a maximum edge capacity and a travel time. For example, at edge N4-N6, the maximum edge capacity is 5, which means at each time point, at most 5 evacuees can start to travel from node N4 to N6 through this link. The travel time of this edge is 4, which means it takes 4 time units to travel from node N4 to N6. This approach of modeling an evacuation scenario to a capacitated node-edge graph is similar to methods presented in related work listed in [1].

As shown in Figure 3, suppose we initially have 10 evacuees at node N1, 5 at node N2, and 15 at node N8. The task is to compute an evacuation plan that evacuates the 30 evacuees to the two destinations (node N13 and N14) using the least amount of time. Figure 2 (b)shows an example evacuation plan for the evacuation network in Figure 2(a). In the result, each row shows one group of evacuees moving together during the evacuation with a group ID, source node, number of evacuees in the group, the evacuation route with time schedule, and the destination time. The route is shown by a series of node numbers and the time schedule is shown by a start time associated with each node on the route. Take source node N8 for example: initially there are 15 evacuees at N8.



Group of Evacuees			Route with Schedule	Dest. Time
ID	Source	Group Size		
A	N8	6	R8(T0)–N10(T3)–N13	4
B	N8	6	N8(T1)–N10(T4)–N13	5
C	N8	3	N8(T0)–N11(T3)–N14	5
D	N1	3	N1(T0)–N3(T1)–N4(T4)–N6(T8)–N10(T13)–N13	14
E	N1	3	N1(T0)–N3(T2)–N4(T5)–N6(T9)–N10(T14)–N13	15
F	N1	1	N1(T0)–N3(T1)–N5(T4)–N7(T8)–N11(T13)–N14	15
G	N2	2	N2(T0)–N3(T1)–N5(T4)–N7(T8)–N11(T13)–N14	15
H	N2	3	N2(T0)–N3(T3)–N4(T6)–N6(T10)–N10(T15)–N13	16
I	N1	3	N1(T1)–N3(T2)–N5(T5)–N7(N9)–N11(T14)–N14	16

(a) Node-edge graph model of example evacuation network

(b) An example of evacuation plan

Figure 2: An example illustrating the input and output of the evacuation planning problem

They are divided into 3 groups: Group A with 6 people, Group B with 6 people and Group C with 3 people. Group A starts from node N8 at time 0 and travels to node N10, then it travels from node N10 at time 3 to node N13, reaching destination N13 at time 4. Group B follows the same route but has a different schedule due to capacity constraints of this route. This group starts from N8 at time 1 and travels to N10, then travels from N10 at time 4 to N13, and reaches destination N13 at time 5. Group C takes a different route. It moves from N8 at time 0 to N11, then moves from N11 at time 3 to N14, and reaches destination N14 at time 5. The procedure is similar for other groups of evacuees from source node N1 and N2. The whole evacuation egress time is 16 time units since the last groups of people (Groups H and I) reach their destination at time 16. This evacuation plan is an optimal plan for the evacuation scenario shown in Figure 2 (a). According to Hoppe and Tardos [10], this problem can be solved using the ellipsoid method, and the total time complexity will reach $O((T \cdot n)^6)$, where n is the number of nodes in the network and T is the user-provided evacuation time upper bound. This problem is hard also due to the violation of assumptions like the stationary ranking of alternative routes, etc. For example, in Figure 2(b), the best route starting at N8 changed from N8-N10-N13 to N8-N11-N14.

3 The Capacity Constrained Route Planner (CCRP)

In our problem formulation, we allow time dependent node capacity and edge capacity, but we assume that edge capacity does not depend on the actual flow amount in the edge. We also allow time dependent edge travel time, but we require that the network preserve the FIFO (First-In First-Out) property. In this section, we briefly describe the main idea of the CCRP algorithm and its experimental evaluation results.

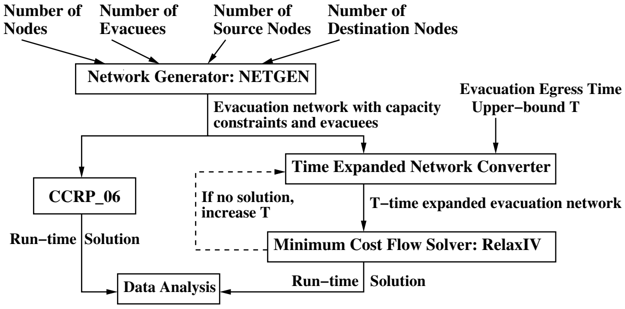


Figure 3: Experiment design

The CCRP Algorithm: The CCRP produces sub-optimal solutions for evacuation planning. We model edge capacity and node capacity as a time series instead of fixed numbers. A time series represents the available capacity at each time instant for a given edge or node, which advances the idea of Time-Aggregated Graph [6]. CCRP uses a heuristic approach based on an extension of shortest path algorithms to account for capacity constraints of the network [1]. The CCRP uses an iterative approach. In each iteration, the algorithm first searches for route R with the earliest destination arrival time from any source node to any destination node, taking previous reservations and possible waiting time into consideration. Next, it computes the actual number of evacuees that will travel through route R. This number is affected by the available capacity of route R and the remaining number of evacuees. Then, it reserves the node and edge capacity on route R for those evacuees. The algorithm continues to iterate until all evacuees reach the destination. The cost model of the above CCRP algorithm is $O(p \cdot nlogn)$, where n is the number of nodes and p is the number of evacuees. A formalized algorithm and detailed analysis can be found in the paper by Lu et al. [1]. In our later work, the CCRP was improved by adding a super node which connects all the source nodes by edges with infinite capacity and zero travel time [9]. This allows us to complete the search for route R by using only one single generalized shortest path search starting from the super node. The run time of the algorithm is reduced most when using Dijkstra's algorithm to search the shortest path.

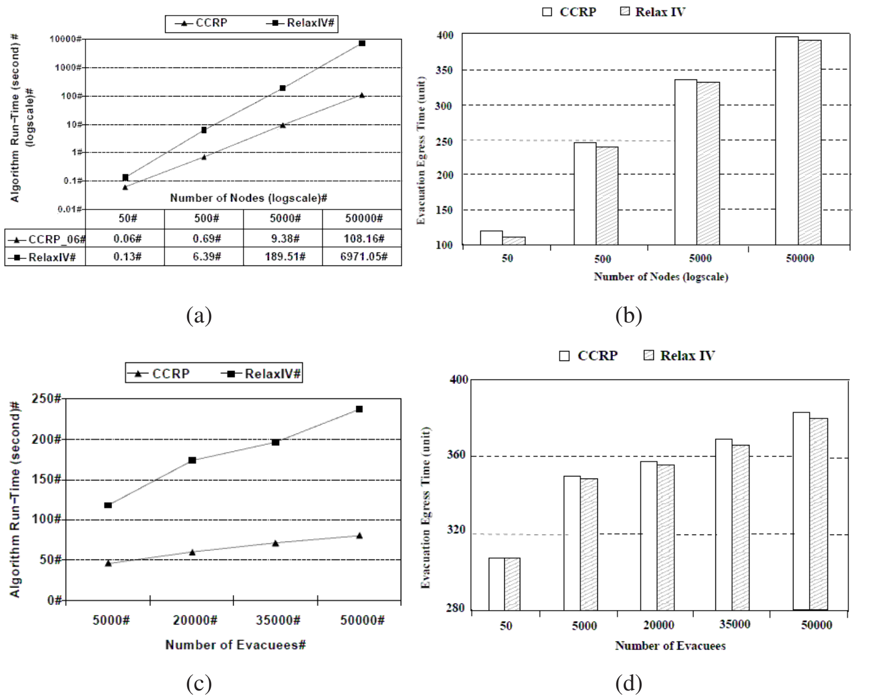


Figure 4: Evaluation results of CCRP: (a) Run-time (log-scale) with respect to network size; (b) Result quality(total evacuation time) with respect to the number of nodes in the network; (c) Run-time with respect to the number of evacuees; (d) Result quality with respect to the number of evacuees.

Experimental Evaluation: We tested the CCRP (the improved algorithm) with different settings and compared its performance with the linear programming-based system Relax IV [7]. Figure 3 shows the experiment design (CCRP_06 refers to the improved CCRP algorithm). First, we studied how the number of nodes in the network affects the run-time cost and the result quality (total evacuation time). The network generated had 5000 evacuees, 20 source nodes and 10 destination nodes. The total number of nodes ranged from 50 to 50000. Results showed that the time cost of CCRP, compared to Relax IV, is not only faster but also grows more slowly as the network size increases (See Figure 4 (a)). Meanwhile, the result quality of CCRP is almost the same as Relax IV (See Figure 4(b)). Then we tested how the number of evacuees affects the run-time and the result quality. The network had 5000 nodes with 2000 sources. The number of evacuees ranged from 5000 to 50000. CCRP produced almost as good result as Relax IV, with much less time spent dealing with increasing numbers of evacuees (See Figure 4(c) and (d)). We also tested the impact of the number of sources and destinations on the time cost and the result quality (see [9]). Results showed that CCRP can scale up to large networks, and can always produce high quality results with less time cost than Relax IV. The experiments were conducted on a workstation with Intel Pentium 4 2.8GHz CPU, 2GB RAM and Linux operating system. Each experimental result shown is the average over 5 experiments runs with networks generated using the same parameters.

4 Case Study and Social Impact

In this section, we report the case study results of CCRP in a major metropolitan region evacuation planning.

Evacuation Planning: Monticello Power Plant: As shown in Figure 5, the Monticello nuclear power plant is located about 40 miles to the northwest of the Twin Cities of Minneapolis-St.Paul. Evacuation plans need to be in place in case of accidents or terrorist attacks. The evacuation zone is a 10-mile radius around the nuclear power plant as defined by Minnesota Homeland Security and Emergency Management [9]. An initial hand-drafted evacuation route plan called for the affected population to a nearby high school. However, this plan did not consider the capacity of the road network and put high loads on two highways.

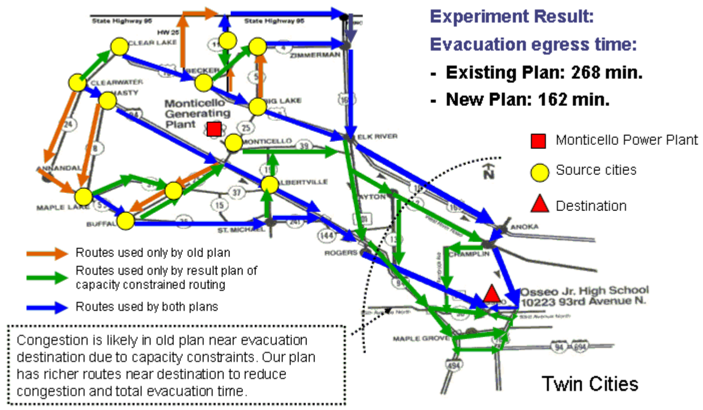


Figure 5: Overlay of result routes for Monticello power plant evacuation route planning. (Best in color)

We experimentally tested the CCRP algorithm using the road network around the evacuation zone provided by the Minnesota Department of Transportation [9] and the Census 2000 population data for each affected city (circle in Figure 5. The total number of evacuees was about 42,000. As can be seen in Figure 5, our algorithm produced a much better evacuation route plan a) by selecting shorter paths to reduce evacuation time, and b) by utilizing richer routes (routes near the evacuation destination) to reduce congestion. As a result, evacuation egress time was reduced from 268 minutes under the old plan to only 162 minutes with CCRP. This experiment demonstrated the effectiveness of our algorithm in real evacuation planning scenarios to reduce evacuation time and improve existing plans.

Evacuation Planning System for Twin Cities Metro Area: Our method was also selected by the Minnesota Department of Transportation to be used in an evacuation planning project for the entire Twin Cities Metro Area, a region with a road network of about 250,000 nodes and a population of over 2 million people. In this project, the CCRP algorithm was incorporated into an evacuation planning system, and was tested on five pre-defined scenarios and some randomly selected locations. This system has several common settings and functions, such as identifying bottleneck areas and links, and designing/refining transportation networks. Particularly useful are the compare options where users can compare the effect of transportation modes (walking and driving percentage) and Time (day-time and night-time) on population distribution. The settings we used in the five pre-defined scenarios are shown in Table 1. One especially interesting finding was that walking results in less congestion than driving, and thus can decrease total egress time. Transportation professionals evaluated the quality of the results and found them to be highly satisfactory.

Social Impact and Public Recognition: Our approach was presented at the Congressional Breakfast Program on Homeland Security held by the University Consortium for Geographic Information Science (UCGIS), and also reported in the Minnesota Homeland Security and Emergency Management newsletter. In addition, this work received the 2006 CTS award [11]. The research results are also reported by Fox TV [8], Pioneer Press, Star Tribune and other local and university media.

5 Extension and Future Work

We extended CCRP by adding contraflow reconfiguration in a later paper [12] which employs lane reversal as a method to reduce congestion by increasing capacity of roads. In the future, work is needed in a number of other areas. For spatial network databases, efficient data models and algorithms are needed for queries with time-variants in flow networks(see Table 2). Regarding emergency management, one of the limitations of current work is that it assumes constant travel time of edges in the traffic network model. In real emergencies, this assumption is sometimes violated due to dense traffic or road damage. We plan to work on improving our current methods to deal with dynamic network settings. Another challenge arises from the multi-criterion feature of evacuation planning, which means that we may have multiple objects to optimize (e.g., food supply sufficiency, equatability, etc) rather than merely the egress time when designing evacuation plans. Since a global optimal plan is usually

Scenario	Source Radius	Destination Radius	Population	Evacuation Time
A	1 mile	1 mile	148,077	4 hrs 46 min
B	1 mile	1 mile	84,678	2 hrs 44 min
C	1 mile	1 mile	27,406	4 hrs 27 min
D	1 mile	1 mile	49,800	3 hrs 39 min
E	1 mile	1 mile	2,586	1 hr 20 min

Table 1: Settings and results of the five scenarios in Twin Cities Metro Area evacuation planning

difficult to find, how to efficiently generate a complete and correct candidate set that does not include any plans worse than any existing plan on all objectives is a challenging problem that we will investigate in the future.

	Static	Time-Variant
Graph (No capacity constraints)	Which is the shortest travel time path from downtown Minneapolis to the airport?	Which is the shortest travel time path from downtown Minneapolis to airport at different times of a work day?
Flow Network	What is the capacity of Twin-Cities freeway network to evacuate downtown Minneapolis?	**What is the capacity of Twin-Cities freeway network to evacuate downtown Minneapolis at different times in a work day?**

Table 2: Example queries with time-variance and flow networks.

Acknowledgements: This work is supported by USDOD Grants HM1582-08-1-0017, HM1582-07-1-2035 and W9132V-09-C-0009, and NSF Grants NSF III-CXT IIS-0713214, NSF IGERT DGE-0504195 and NSF CRI:IAD CNS-0708604.

References

[1] Q. Lu, B. George, and S. Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. *Proceedings of the 9th International Symposium on Spatial and Temporal Databases (SSTD'05)*, pages 291-307, 2005. Springer Verlag.

[2] H.S. Mahmassani, H. Sbayti, and X. Zhou. DYNASMART-P Version 1.0 Users Guide. *University of Maryland*, Maryland, 2004.

[3] J. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 2(1), 1952. Thomas Telford Ltd.

[4] The White House, Federal response to hurricane katrina: Lessons learned. `http://www.whitehouse.gov/reports/katrina-lessons-learned/`, 2006.

[5] H.W. Hamacher and S.A. Tjandra. Mathematical modelling of evacuation problems - a state of the art. *Pedestrian and Evacuation Dynamics*, pages 227-266, 2002. Springer Verlag.

[6] B. George, S. Kim, and S. Shekhar. Spatio-temporal network databases and routing algorithms: A summary of results.*Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD'07)*, pages 460-477, 2007. Springer Verlag.

[7] D.P. Bertsekas and P. Tseng. RELAX-IV: A faster version of the RELAX code for solving minimum cost flow problems, Technical Report P-2276, Massachusetts Institute of Technology, 1994.

[8] Fox TV News. Evening News (May 8th, 2006), `http://www.cs.umn.edu/shekhar/talk/video/fox9airedmpg.avi`.

[9] Q. Lu, B. George, and S. Shekhar. Evacuation Route Planning: A Case Study in Semantic Computing. *International Journal of Semantic Computing*, 1(2):249, 2007. World Scientific Publishing Co..

[10] B. Hoppe and E. Tardos: Polynomial Time Algorithms For Some Evacuation Problems. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, page 433-441, 1994. SIAM.

[11] University of Minnesota Center for Transportation Studies. Evacuation Project Wins Award, CTS Report (May, 2006). `http://www.cts.umn.edu/news/report/2006/05/EvacuationProject.html`, 2006.

[12] S. Kim, S. Shekhar, and M. Min. Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1115-1129, 2008.

GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory

Yu Zheng, Xing Xie and Wei-Ying Ma
Microsoft Research Asia, 4F Sigma Building, NO. 49 Zhichun Road, Beijing 100190, China
{yuzheng, xingx, wyma}@microsoft.com

Abstract

People travel in the real world and leave their location history in a form of trajectories. These trajectories do not only connect locations in the physical world but also bridge the gap between people and locations. This paper introduces a social networking service, called GeoLife, which aims to understand trajectories, locations and users, and mine the correlation between users and locations in terms of user-generated GPS trajectories. GeoLife offers three key applications scenarios: 1) sharing life experiences based on GPS trajectories; 2) generic travel recommendations, e.g., the top interesting locations, travel sequences among locations and travel experts in a given region; and 3) personalized friend and location recommendation.

1 INTRODUCTION

The advance of location-acquisition technologies like GPS and Wi-Fi has enabled people to record their location history with a sequence of time-stamped locations, called trajectories. These trajectories imply to some extent users' life interests and preferences, and have facilitated people to do many things, such as online life experience sharing [12, 13], sports activity analysis [4] and geo-tagging multimedia content [3]. Using these trajectories, we cannot only connect locations in the physical world but also bridge the gap between users and locations and further deduce the connections among users. That is, we are able to understand both users and locations based on the trajectories.

In this paper, we introduce our project GeoLife [1, 2, 5 14], which is a social networking service incorporating users, locations and user-generated GPS trajectories. As demonstrated in the left part of Figure 1, people access a sequence of locations in the real world and generate many trajectories in a form of GPS logs. Based on these GPS trajectories, we can build three graphs: a location-location graph, a user-location graph, and a user-user graph. In the location-location graph, a node is a location and a directed edge between two locations stands for that a least some users have consecutively traversed these two locations in a trip. In the user-location graph, there are two types of nodes, users and locations. An edge starting from a user and ending at a location indicates that the user has visited this location for some times, and the weight of the edge is the visiting times of the user. Further, we can infer the user-user graph where a node is a user and an edge between two nodes represents that the two users have visited the same location in the real world for some times (the edge weight is the times sharing the same locations).

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

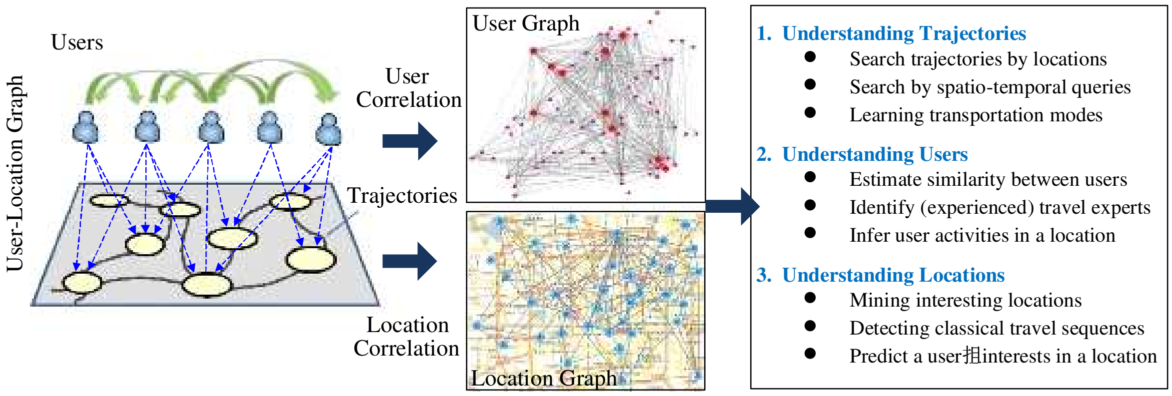


Figure 1: The philosophy and research points of GeoLife

Using the graphs mentioned above, we conduct three aspects of research listed in the right part of Figure 1.
1) *Understanding trajectories*: For example, we provide a method to search for the trajectories by giving a set of locations of interests [16], or by issuing a spatial query range combined with a temporal interval [1] as a query. Meanwhile, we infer the transportation modes that a user took when generating a GPS trajectory [8, 11, 15].
2) *Understand users*: First, we estimate the similarity between each pair of users in terms of their location histories represented by GPS trajectories [2]. Second, using an iterative inference model, we compute the travel experience of each user based on their location history, and then find out travel experts in a geo-region by ranking users according to the experiences [9]. Third, we can infer the activities that a user can perform in a location based on multiple users' location histories and the associated comments [5][6].
3) *Understanding locations*: On example is that we mine the top interesting locations and travel sequences in a given region from a large number of users' GPS trajectories. Another example is we predict a user's interests in an unvisited location by involving the GPS trajectories of the user and that of others.

In the rest of this paper, we present three application scenarios, each of which includes several research points mentioned above. These scenarios are 1) sharing life experiences based on GPS trajectories, 2) generic travel recommendation, and 3) personalized friend and travel recommendation.

2 SHARE LIFE EXPERIENCES BASED ON GPS TRAJECTORIES

Recently, a branch of GPS-trajectory-sharing applications [3, 4, 12, 13] appeared on the Internet. In these applications, people can record their travel routes using a GPS-enabled device and then share travel experiences among each other by publishing these GPS trajectories in a Web community. Photos, comments and tips can also be associated with the related locations in a trajectory. GPS-trajectories-sharing offers a more fancy and interactive approach than text-based articles to better express people's travel experiences, which provide users with valuable references when planning a travel itinerary. However, as a large-scale GPS tracks have been accumulated, how to manage these GPS data is an important issue in these applications. Obviously, users need an efficient approach to retrieve the specific GPS trajectories they are interested in, since nobody has time and patience to browse trajectories one-by-one. Moreover, people intend to learn information about user behaviors as well as user intentions behind the raw data. Thus, this section first presents two approaches that can facilitate users to search for trajectories efficiently. Then, we introduce a machine-learning algorithm inferring the transportation modes of a GPS trajectory. This technique can provide us with deep understanding of a trajectory and achieve the automatic categorization of GPS trajectories by transportation modes, and further enable trajectory filtering by transportation modes.

2.1 Search Trajectories by Location

When traveling to an unfamiliar city, people usually have a set of places of interests in their mind while they do not know how to travel among these places. For example, most of people visiting Beijing know or hear of Tiananmen Square, the Summer Palace and the Bird's-Nest in the Olympic Park. However, they have no idea how to travel in each place and what a proper visiting sequence among these places is. At this moment, a user can give the names of these three locations as a query, or directly point out the locations of these places on a map (if they know where these places are). Then, our method searches the existing database for some trajectories that were generated by other users and traverse these places. In short, this query can benefit travelers planning a trip to multiple places of interest in an unfamiliar city by providing similar routes traveled by other people.

In this work, we propose a new type of trajectory query called the k Best-Connected Trajectory (k-BCT) query for searching trajectories by multiple geographical locations. Generally, the k-BCT query is a set of locations indicated by coordinates like (latitude, longitude), which could be a famous attraction, a nameless beach, or any arbitrary place approximated by the center location. The user may also specify a preferred visiting order for the intended places, in which case the order of a trajectory needs to be taken into account as well. To answer the k-BCT query, we first define a similarity function, which should consider the distance from the trajectory to each query location, to measure how well a trajectory connects the query locations. Second, we propose an Incremental k-NN based Algorithm (IKNN), which retrieves the nearest trajectory points with regard to each query location incrementally and examines the k-BCT from the trajectory points discovered so far. In this algorithm, the pruning and refinement of the search are conducted by using the lower bound and upper bound of trajectory similarity that are derived from the found trajectory points. The retrieval of nearest trajectory points is based on the traditional best-first and depth-first k-NN algorithms over an R-tree index. Refer [16] for details.

2.2 Search Trajectories by Spatio-Temporal Queries

Sometimes, people are interested in some GPS trajectories showing the travel experiences within a particular geo-region and in a specific time interval. For instance, what is fun in the Olympic Park of Beijing in the weekend? Where would be interesting in the downtown Beijing during Christmas? These questions can be represented by a spatio-temporal query, with which we can retrieve some existing trajectories providing reference travel experiences. In these online GPS-trajectory-sharing applications, we observe that users tend to upload GPS trajectories of the near past more frequently than the trajectories of the distant past. For instance, users are more likely to upload GPS trajectories of today or yesterday than those of months ago. Thus, traditional spatio-temporal indexing schemes, like R tree or its variants, are not optimal to handle the skewed nature of accumulative GPS trajectories.

We propose Compressed Start-End Tree (CSE-tree) [1] for the GPS data sharing applications based on users' uploading behavior. In this scheme, we first partition the space into disjoint cells that cover the whole spatial region, and then maintain a flexible temporal index for each spatial cell. To insert a new GPS track, we divide the track into segments according to spatial partition. Then each segment is inserted into the temporal index of corresponding spatial grid. For all segments in a temporal index, they are divided into several groups according to end time of the segment. We observed that for different groups, the frequency of new updates is different. CSE-tree uses B+ tree index for frequently updated groups and sorted dynamic array for rarely updated ones. The update frequency of a group may change as time goes by, so we transform a B+ tree into a sorted dynamic array if update frequency of a group drops below a threshold. Our contribution lies in the following two aspects.

- A stochastic process model is proposed to simulate user behavior of uploading GPS tracks to online sharing applications. This model can also be applied to other data sharing applications on the Web.
- A novel indexing scheme is optimized to the user behavior of uploading GPS tracks. Our scheme requires less index space and less update cost while keeping satisfactory retrieval performance.

2.3 Learning transportation modes based on GPS Data

As a kind of human behavior, people's transportation modes, such as walking and driving, can provide pervasive computing systems with more contextual information and enrich a user's mobility with informative knowledge. With the transportation modes of a GPS track, people can obtain more reference knowledge from others' trajectories. Users can know not only where other people have been but also how these people reach each location. Meanwhile, such knowledge enables the classification of GPS trajectories by transportation modes. So, we can perform smart route recommendations/designs for a person based on the person's needs. For instance, a system should return a bus line rather than a driving route to an individual intending to move to somewhere by a bus.

We designed an approach based on supervised learning to automatically infer users' transportation modes, including driving, walking, taking a bus and riding a bike, from raw GPS logs. Our approach consists of three parts: a change point-based segmentation method, an inference model and a graph-based post-processing algorithm. First, we propose a change point-based segmentation method to partition each GPS trajectory into separate segments of different transportation modes. The key insight of this step is that people need to walk for a while before transferring to another transportation mode. Second, from each segment, we identify a set of sophisticated features, which are not affected by differing traffic conditions (e.g., a person's direction when in a car is constrained more by the road than any change in traffic conditions). Later, these features are fed to a generative inference model to classify the segments of different modes. Third, we conduct graph-based post-processing to further improve the inference performance. This post-processing algorithm considers both the commonsense constraints of the real world and typical user behaviors based on locations in a probabilistic manner. Refer to papers [8, 11, 15] for details.

The advantages of our method over the related works include three aspects. 1) Our approach can effectively segment trajectories containing multiple transportation modes. 2) Our work mined the location constraints from user-generated GPS logs, while being independent of additional sensor data and map information like road networks and bus stops. 3) The model learned from the dataset of some users can be applied to infer GPS data from others. Using the GPS logs collected by 65 people over a period of 10 months, we evaluated our approach via a set of experiments. As a result, based on the change-point-based segmentation method and Decision Tree-based inference model, we achieved prediction accuracy greater than 71 percent. Further, using the graph-based post-processing algorithm, the performance attained a 4-percent enhancement.

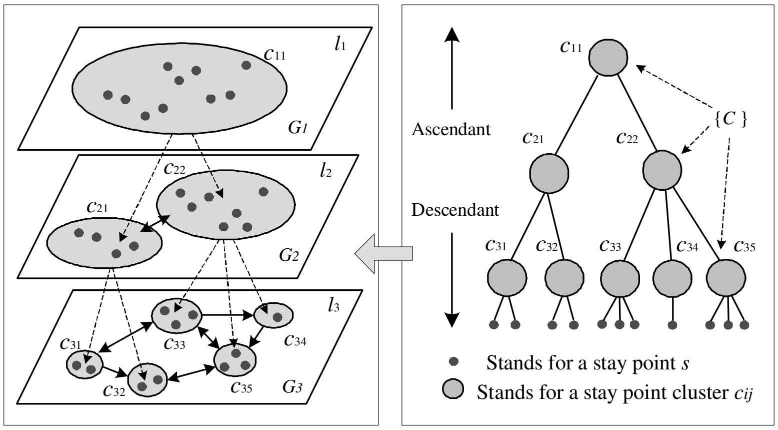
3 GENERIC TRAVEL RECOMMENDATION

3.1 Mining Travel Experts, Interesting Locations and Travel Sequences

This recommender provides a user with the top n experienced users (experts), interesting locations and the classical travel sequences among these locations, in a given geospatial region. To define interesting location, we mean the culturally important places, such as Tiananmen Square in Beijing and the Statue of Liberty in New York (i.e. popular tourist destinations), and commonly frequented public areas, such as shopping malls/streets, restaurants, cinemas and bars. With the information mentioned above, an individual can understand an unfamiliar city in a very short period and plan their journeys with minimal effort.

However, we will meet two challenges when conducting this recommendation. First, the interest level of a location does not only depend on the number of users visiting this location but also lie in these users' travel experiences. Intrinsically, different people have different degrees of knowledge about a geospatial region. For example, the local people of Beijing are more capable than overseas tourists of finding out high quality restaurants and famous shopping malls in Beijing. Second, an individual's travel experience and interest level of a location are relative values (i.e., it is not reasonable to judge whether or not a location is interesting), and are region-related (i.e., conditioned by the given geospatial region). A user, who has visited many places in a city like New York, might have no idea about another city, such as Beijing.

To achieve this generic recommendation, we first propose a tree-based hierarchical graph (TBHG) that models multiple users' travel sequences on a variety of geospatial scales. As demonstrated in Figure 2, three steps need to be performed when building a TBHG. 1) Detect stay points: We detect from each GPS trajectory some stay points [2] where a user has stayed in a certain distance threshold over a time period. 2) Formulate a tree-based Hierarchy H: We put together the stay points detected from users' GPS logs into a dataset. Using a density-based clustering algorithm, we hierarchically cluster this dataset into some geospatial regions (a set of clusters C) in a divisive manner. Thus, the similar stay points from various users would be assigned to the same clusters on different levels. 3) Build graphs on each level: Based on the tree-based hierarchy H and users' location histories, we can connect the clusters of the same level with directed edges. If consecutive stay points from one trip are individually contained in two clusters, a link would be generated between the two clusters in a chronological direction according to the time serial of the two stay points.



A Tree-Based Hierarchical Graph A Tree-Based Hierarchy

Figure 2: Building a tree-based hierarchical graph

Based on the TBHG, we propose a HITS-based model to infer users' travel experiences and interest of a location within a region. This model leverages the main strength of HITS to rank locations and users with the context of a geospatial region, while calculating hub and authority scores offline. Therefore, we can ensure the efficiency of our system while supporting users to specify any georegions as queries. Using the third level of the TBHG shown in Figure 2 as a case, Figure 3 illustrates the main idea of our HITS-based inference model. Here, a location is a cluster of stay points, like c_{31} and c_{32}. We regard an individual's visit to a location as an implicitly directed link from the individual to that location. For instance, cluster c_{31} contains two stay points respectively detected from u_1 and u_2's GPS traces, i.e., both u_1 and u_2 have visited this location. Thus, two directed links are generated respectively to point to c_{31} from u_1 and u_2. Similar to HITS, in our model, a hub is a user who has accessed many places, and an authority is a location which has been visited by many users. Therefore, users' travel experiences (hub scores) and the interests of locations (authority scores) have a mutual reinforcement relation. More specifically, a user's travel experience can be represented by the interest levels of the visited locations. In turn, the interest level of a location can be calculated by the experiences of the users who have accessed this location. Using a power iteration method, we can generate the final score for each user and location, and find out the top n interesting locations and experience users in a given region.

With users' travel experiences and the interests of locations, we calculate a classical score for each location sequence within the given geospatial region. The classical score of a sequence is the integration of the following three aspects. 1) The sum of hub scores of the users who have taken this sequence. 2) The authority scores of the locations contained in this sequence. 3) These authority scores are weighted based on the probability that people would take a specific sequence. Refer to papers [9, 14] for details.

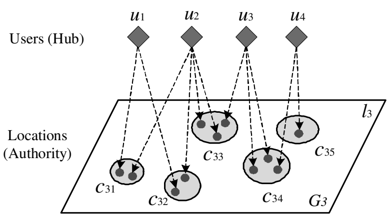


Figure 3: Our HITS-based inference model

3.2 Collaborative Location and Activity Recommendation

Typically, people would have the following two types of questions in their mind when traveling. One is that, if we want to do something like sightseeing or food-hunting in a large city, where should we go? The other is, if we have already visited some places, such as the Olympic park of Beijing, what else can we do there? In general, the first question corresponds to location recommendation given some activity query (where "activity" can refer to various human behaviors such as food-hunting, shopping, watching movies/shows, enjoying sports/exercises, tourism, etc.), and the second question corresponds to activity recommendation given some location query.

In this work, for the first question, we can provide a user with a list of interesting locations, such as Tiananmen Square and Bird's Nest. For the second question, if a user visits Bird's Nest, we can recommend her to not only go sightseeing but also experience some outdoor exercise facilities or try some nice food nearby. We put both location recommendation and activity recommendation together in the knowledge mining, since locations and activities are closely related in nature. As we mentioned, to better share life experience, a user can add some comments or tips to a point location in a trajectory. Such comments and tips imply a user's behavior and intentions when visiting a location. With the available user comments, we can get the statistics about what kinds of activities the users performed on a location, and how often they performed these activities. As shown in the middle part of Figure 4, by organizing this statistics' data in a matrix form, we can have a location-activity matrix, with rows as locations and columns as activities. An entry in the matrix denotes the frequency for the users to perform some activity on some location. However, the location-activity matrix is incomplete and very sparse. Therefore, we leverage the information from another two matrices, the location-feature and activity-activity matrices, respectively shown in the left and right part of Figure 4.

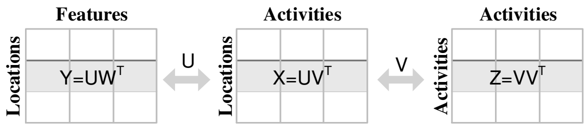


Figure 4: Collaborative location-activity learning model

Location-feature matrix: We exploit the location features with the help of POI category database. The database is based on the city yellow pages, and it can provide us the knowledge that what kinds of POIs we have in an area. This helps us to get some sense of this location's functionalities, so that we can use them as features for better recommendations. Similarly, by organizing the data in a matrix form, we can have a location-feature matrix, where each entry of the matrix denotes some feature value on that location.

Activity-activity matrix: We exploit the World Wide Web to get the knowledge about the activity correlations. With this knowledge, we may better infer that if a user performs some activity on a location, then how likely she would perform another activity. By organizing the data in a matrix form, we have an activity-activity matrix, where each entry of the matrix denotes the correlation between a pair of activities.

After the data modeling, we have the location-activity, location-feature and activity-activity matrices. Our objective is to fill those missing entries in the location-activity matrix with the information learned from the other two matrices, so as to get a full matrix for both location and activity recommendations. Here, we provided a collaborative filtering (CF) approach based on collective matrix factorization to take these information sources as inputs and train a location and activity recommender. By defining an objective function and using the gradient descent to iteratively minimize the objective function, we can infer the value of each missed entry. Based on the filled location-activity matrix, we can rank and retrieve the top k locations/activities as recommendations to the users. Refer to papers [5, 6] for details.

4 PERSONALIZED FRIEND & LOCATION RECOMMENDATION

Besides the generic recommendation, an individual also wants to visit some locations matching her travel preferences (personalized). Actually, people's outdoor movements in the real world would imply rich information about their life interests and preferences. For example, if a person usually goes to stadiums and gyms, it denotes that the person might like sports. According to the first law of geography, everything is related to everything else, but near things are more related than distant things, people who have similar location histories might share similar interests and preferences. The more location histories they share, the more correlated these two users would be. Therefore, based on users' GPS trajectories we conduct a personalized friend & location recommender, which provides an individual with some similar users in terms of location histories and recommends some places that could interest the individual while having not been found by the individual.

4.1 Friend Recommendation

To achieve the friend recommendation, we first build a shared framework shown in Figure 2 according to the first two steps mentioned in Section 3.1. By feeding each user's GPS trajectories into this framework, we can build a personal hierarchical graph for each user. Based on this graph, we propose a similarity measurement, referred to as a hierarchical-graph-based similarity measurement (HGSM), which takes into account the following three factors to estimate the similarity between users:
1) *The sequence property of people's outdoor movements*: The longer similar sequences matched between two users' location histories, the more related these two users might be.
2) *The hierarchical property of geographic spaces*: Users who share similar location histories on geographical spaces of finer granularities might be more correlated.
3) *The visited popularity of a location*: Similar to inverse document frequency, two users who accessed a location visited by a few people might be more correlated than others sharing a location history accessed by many people.

Given two users' hierarchical graphs, we can calculate a similarity score for them by using the HGSM. Later, a group of people, called potential friends, with relatively high scores will be retrieved for a particular individual. Refer to paper [2] for details.

4.2 Personalized Location Recommendation

This recommender uses a particular individual's visits on a geospatial location as their implicit ratings on the location, and predicts a particular user's interest in an unvisited location in terms of their location history and those of other users. To achieve this recommender, we first formulate a matrix between users and locations, where rows stand for users and columns represent users' ratings on locations. We incorporated a content-based method into a user-based collaborative filtering algorithm, which uses HGSM as the user similarity measure, to estimate the rating of a user on an item. Later, some unvisited locations that might match their tastes can be recommended to the individual. Refer to paper [10] for details. Though the CF model using HGSM as a similarity measurement is more effective than those using the Pearson correlation and the Cosine similarity, it consumes a lot of computation since the HGSM-based method needs to calculate the similarity between each pair of users while the number of users could keep on increasing as a system becomes popular. To address this problem, we propose an item-based CF model regarding locations as items. In this work, we first mine the correlation among locations from multiple users' GPS traces in terms of 1) the sequences that the locations have been visited and 2) the travel experiences of the users creating these sequences. Then, the location correlation is incorporated into a CF-based model that infers a user's interests in an unvisited location based on her locations histories and that of others. The item-based CF model using location correlation is slightly less effective than the HGSM-based one while is much more efficient than the latter. Papers [7, 14] offer details.

5 CONCLUSION

User-generated GPS trajectories do not only connect locations in the physical world but also bridge the gap between people and locations. In GeoLife, we aim to understand trajectories, users, and locations in a collaborative manner, and perform three key application scenarios. The first scenario, sharing life experience based on GPS trajectory, focuses on understanding GPS trajectories. In the second scenario, generic travel recommendation, we infer the travel experience of a user (understand users) and the interest level of a location (understand locations) in an iterative manner. Later, the classical travel sequences among locations (location correlation) are detected based on the inferred user experience and location interest. Meanwhile, we learn user activities in a location (user-location correlation) and enable a location-activity recommender. In the third scenario, personalized friend and location recommendation, we estimate the similarity between users (user correlation) based on the similarity between their location histories (location correlation). Later, a personalized recommender, which predicts a user's interests in an unvisited location (user-location correlation) by integrating this user similarity into a CF model, is conducted. Overall, user, location and trajectory have a collaborative and mutual reinforcement relationship among each other.

References

[1] Longhao Wang, Yu Zheng, Xing Xie, Wei-Ying Ma. A Flexible Spatio-Temporal Indexing Scheme for Large-Scale GPS Track Retrieval, In MDM 2008.

[2] Quannan Li, Yu Zheng, Yukun Chen, Xing Xie. Mining user similarity based on location history. In ACM SIGSPATIAL GIS 2008.

[3] Scott Counts, M. Smith. Where were we: Communities for sharing space-time trails. In ACM GIS 2007

[4] SPORTSDO. 2007. http://sportsdo.net/Activity/ActivityBlog.aspx.

[5] Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, Qiang Yang. Collaborative Filtering Meets Mobile Recommendation: A User-centered Approach. In AAAI 2010.

[6] Wencheng Zheng, Yu Zheng, Xing Xie, Qiang Yang. Collaborative Location and Activity Recommendations With GPS History Data. In WWW 2010.

[7] Yu Zheng, Xing Xie. Learning Location Correlation from User-Generated GPS trajectories. In MDM 2010.

[8] Yu Zheng, Like Liu, Longhao Wang, Xing Xie. Learning Transportation Modes from Raw GPS Data for Geographic Application on the Web, In WWW 2008.

[9] Yu Zheng, Lizhu Zhang, Xing Xie, Wei-Ying Ma. Mining interesting locations and travel sequences from GPS trajectories. In WWW 2009

[10] Yu Zheng, Lizhu Zhang, Xing Xie. Recommending friends and locations based on individual location history. To appear in ACM Transaction on the Web, 2010.

[11] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie. Understanding Mobility Based on GPS Data. In UbiComp 2008.

[12] Yu Zheng, Yukun Chen, Xing Xie, Wei-Ying Ma. GeoLife2.0: A Location-Based Social Networking Service. In MDM 2009.

[13] Yu Zheng, Longhao Wang, Xing Xie, Wei-Ying Ma. GeoLife-Managing and understanding your past life over maps, In MDM 2008.

[14] Yu Zheng, Xing Xie. Learning travel recommendation from user-generated GPS trajectories. To appear in ACM Transaction on Intelligent Systems and Technologies (ACM TIST).

[15] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, Wei-Ying Ma. Understanding transportation modes based on GPS data for Web applications. ACM Transaction on the Web. 4(1), January, 2010. 1-36.

[16] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, Xing Xie. Searching Trajectories by Locations - An Efficiency Study, In ACM SIGMOD 2010

GeoSIM: A Geospatial Data Collection System for Participatory Urban Texture Documentation

Farnoush Banaei-Kashani Houtan Shirani-Mehr Bei Pan Nicholas Bopp
Luciano Nocera Cyrus Shahabi

Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781
[banaeika, hshirani, beipan, boppnick, nocera, shahabi]@usc.edu

Abstract

Participatory texture documentation (PTD) is a geospatial data collection process in which a group of users (dedicated individuals and/or general public) with camera-equipped mobile phones participate in collaborative collection of urban texture information. PTD enables inexpensive, scalable and high resolution data collection for urban texture mapping. In this paper, we introduce GeoSIM (Geospatial Social Image Mapping), a system we have designed and developed to enable efficient PTD. GeoSIM deploys a two-step planning process for efficient PTD. At the first step, termed "viewpoint selection", a minimum number of points in the urban environment are selected from which the texture of the entire urban environment (the part accessible to cameras) can be collected/captured. At the second step, called "viewpoint assignment", the selected viewpoints are assigned to the participating users such that given a limited number of users with various user constraints (e.g., specific participation time) users can collectively capture maximum amount of texture information within a limited time interval. Viewpoint selection and viewpoint assignment are both NP-hard problems. We present the design and implementation of GeoSIM based on our proposed heuristics for efficient viewpoint selection and viewpoint assignment that enable on-the-fly planning for PTD.

1 Introduction

The advent of earth visualization tools (e.g., Google Earth™, Microsoft Virtual Earth™) has inspired and enabled numerous applications. Some of these tools already include *texture* in their representation of the urban environment. The urban texture consists of the set of images/photos collected from the real environment, to be mapped on the façade of the 3D model of the environment (e.g., building and vegetation models) for photo-realistic 3D representation. Currently, urban texture is collected via aerial and/or ground photography (e.g., Google Street View). As a result, texture collection/documentation is 1) expensive, 2) unscalable (in terms of the required resources), and 3) with low temporal and/or spatial resolution (i.e., texture cannot be collected frequently and widely enough).

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

These limitations can be addressed by leveraging the popularity of camera-equipped mobile devices (such as cell phones and PDAs) for inexpensive and scalable urban texture documentation with high spatiotemporal resolution. With *participatory texture documentation*, termed *PTD* hereafter, a group of participants (dedicated individuals and/or general public) with camera-equipped mobile phones participate in collaborative/social collection of the urban texture information. By enabling low-cost, scalable, accurate, and real-time texture documentation, PTD empowers various applications such as eyewitness news broadcast, urban behavior analysis, real-estate monitoring, emergency-response, and disaster management (e.g., for damage assessment in case of earthquake, hurricane, and wildfire).

With this paper, we introduce *GeoSIM* (*Geospatial Social Image Mapping*), a system we have designed and developed to enable efficient PTD [1]. GeoSIM plans for efficient PTD by employing a two-step process. At the first step, called *viewpoint selection*, a set of points in the urban environment is selected from which the texture information of the entire environment (the part accessible to cameras) can be collected. We call such points as *viewpoints*. Due to the participatory nature of PTD, available resources (e.g., users' participation time) are usually limited and, therefore, it is critical to minimize the number of selected viewpoints. At the second step, termed *viewpoint assignment*, the selected viewpoints are assigned to the users for texture collection. The viewpoints must be assigned such that the texture collected during the documentation campaign (i.e., the specific time interval allocated for texture documentation) is maximized while all users' constraints are satisfied. In [7] and [6], we prove that the problems of viewpoint selection and viewpoint assignment are both NP-hard problems by reduction from the minimum set-cover problem and the team orienteering problem, respectively. Therefore, optimal implementations of viewpoint selection and viewpoint assignment are unscalable and fail to satisfy the real-time requirements of planning for efficient PTD given its participatory nature. Accordingly, we propose efficient heuristics for each of the two problems that are scalable and allow for on-the-fly planning for PTD. GeoSIM is designed and implemented based on our proposed heuristics.

The rest of this paper is organized as follows. In Section 2, we formally define our problem by describing the two-step planning process for efficient PTD. Subsequently, in Section 3 we present our corresponding two-step solution for on-the-fly PTD planning. We discuss the GeoSIM system design and implementation in Section 4. In Section 5, we discuss the related work, and finally, we conclude in Section 6.

2 Problem Definition

In this section, after explaining preliminary concepts, we formally define the viewpoint selection and viewpoint assignment problems in Sections 2.2 and 2.3, respectively.

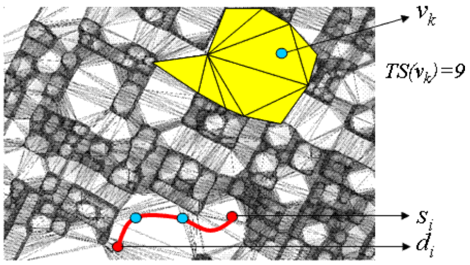
2.1 Preliminaries

Below, first we explain how we model the 3D environment which is subject to texture documentation. Next, we define our assumed user participation model for participatory texture documentation.

2.1.1 Environment Model

Consider an urban environment which consists of various 3D elements such as buildings, trees and terrain (see Figure 1(a), for example). Suppose the environment is modeled in object-level (i.e., a 3D model exists in which the entire environment is represented by a set of objects). Here, without loss of generality, we assume the environment is modeled by the 3D TIN (Triangulated Irregular Network) model. The corresponding TIN model of the environment shown in Figure 1(a) is depicted in Figure 1(b) (shown in 2D). The *texture* of the environment is defined as the set of images mapped on the triangles of the 3D TIN model.

a. Actual Environment



b. TIN Model

Figure 1: 3D Environment Representation for Texture Documentation

2.1.2 Participation Model

A texture documentation campaign is defined as the process of collecting and mapping the environment texture onto the corresponding TIN model of the environment during a predefined time interval T_C, termed the *campaign time* (e.g., 10:00am to 2:00pm on a particular day). We assume the urban texture remains unchanged during T_C. Suppose V is the set of points in the environment such that from each point $v \in V$ one can collect texture information by imaging the surrounding area. We call each such point v a *viewpoint*. Accordingly, we define the *texture score* $TS(v)$ of a viewpoint v as the total number of TIN triangles visible from v. For example, in Figure 1(b) the texture score of the viewpoint v_k is $TS(v_k) = 9$. Similarly, the texture score $TS(W)$ of a set of points $W \subseteq V$ is defined as the total number of TIN triangles visible from any viewpoint in W.

With participatory texture documentation, the texture collection process is implemented by a set of users U. We assume each user $u \in U$ has a set of participation constraints denoted by $c = (s, d, A)$, where s is user's starting point in the environment, d is user's desired ending point (where the user intends to leave the documentation campaign), and A is user's maximum available time for participation. Accordingly, a *participation plan* (or *participation path*) for a user u is define as a path $P_u = (s, v_1, v_2, \ldots, v_n, d)$ that starts from the starting point s and ends at the ending point d while traversing a number of viewpoints v_1 to v_n, where the user is expected to make stops for texture collection. Figure 1(b) shows a sample participation path for a user u_i (not shown in the figure) with the set of constraints $c_i = (s_i, d_i, A_i)$; in this case, the sample participation path traverses two viewpoints. Furthermore, a participation path P_u for user u is said to *satisfy* the user constrains c if and only if the total time to traverse the participation path (i.e., the actual user participation time) is less than the user available time A:

$$t_p + nt_{v_i} \leq A \tag{1}$$

where t_p is the total time to traverse the subpaths between successive viewpoints (assuming shortest path), and t_{v_i} is the time it takes to collect images at each viewpoint v_i along the path P_u. Finally, the texture score $TS(P_u)$ of the path P_u is defined to be equal to the texture score $TS(V_{P_u})$, where V_{P_u} is the set of viewpoints v_1 to v_n covered by P_u. Similarly, the texture score $TS(P_U)$ of a set of paths P_U is defined to be equal to the texture score $TS(V_{P_U})$, where V_{P_U} is the set of viewpoints covered by at least one path in P_U.

2.2 Viewpoint Selection Problem

Suppose T is the set of TIN triangles that comprise the 3D model of the target environment. Consider $T' \subseteq T$ as the subset of TIN triangles that are visible from at least one viewpoint in V (note that given a finite set of viewpoints V, there might be a non-empty set of triangles $T \backslash T'$ that cannot be texture mapped, regardless). Accordingly, we call a set of viewpoints $V' \subseteq V$ a *texture covering* set, if every triangle in T' is visible from at least one viewpoint in V'. The *viewpoint selection* problem is defined as the process of finding a texture covering set V_S with minimum size among all texture covering subsets of V.

2.3 Viewpoint Assignment Problem

Once the texture covering set V_S is identified, the viewpoints $v \in V_S$ must be assigned to the participating users $u \in U$ (by including viewpoints in their participation plan) such that the total number of the TIN triangles $t \in T'$ covered by users within the campaign time T_C is maximized. Formally, the problem of *viewpoint assignment* is defined as an optimization problem, $arg\,max_c \;\; TS(P_U)$, to find the set of participation plans $P_U = \{P_{u_1}, P_{u_2}, \ldots, P_{u_m}\}$ corresponding to the users u_1, u_1,...,u_m in U such that $TS(P_U)$ is maximized while each P_{u_i} satisfies the corresponding user constraints c_i.

With GeoSIM, we assume users can join the texture documentation campaign *progressively* (not necessarily at a single time instant), with a poisson arrival distribution. Accordingly, we generalize the definition of the viewpoint assignment problem by considering an iterative viewpoint assignment scheme. With this scheme, the campaign time T_C is divided into equi-length epochs, I_1 to I_l, and viewpoint assignment is repeated at each epoch to assign the remaining uncovered viewpoints (those viewpoints that are not covered at previous epochs) to the users who arrive within the current epoch I_i. With iterative viewpoint assignment the optimization problem defined above is generalized and modified to $arg\,max_c \;\; TS(P_{U_i})$, where $U_i \subseteq U$ is the subset of users arriving during the i-th epoch I_i.

3 On-the-Fly Planning for Participatory Texture Documentation

We briefly describe our proposed solutions that enable on-the-fly viewpoint selection and viewpoint assignment in Sections 3.1 and 3.2, respectively. The detailed description of our solutions can be found in our extended papers [7] and [6].

3.1 Viewpoint Selection Solution

With [7], we propose an efficient heuristic, termed *GVS* (short for *Greedy Viewpoint Selection*), which allows for approximate but real-time viewpoint selection with approximation guarantees. In essence, GVS is a greedy heuristic that solves a given instance of the viewpoint selection problem by *reduction to* the corresponding instance of the classical minimum set-cover problem. With our experimental results, we show that GVS finds the minimum texture covering set V_S for an area as large as Los Angeles County (covering 1183 square miles with numerous objects) in a few seconds. In this case, V_S only includes 17% of the viewpoints in V.

3.2 Viewpoint Assignment Solution

With [6], we propose two families of efficient heuristics that enable on-the-fly viewpoint assignment: individual-based heuristics and group-based heuristics. With individual-based heuristics, we generate each user participation plan exclusively, independent of those of other users. Toward that end, we reduce the viewpoint assignment problem for a single user to the classical problem of orienteering [3], and accordingly adopt and extend the most recent heuristic solutions for the orienteering problem [4, 5] to implement viewpoint assignment. Individual-based heuristics are efficient and allow for on-the-fly viewpoint assignment; however, due to their exclusive nature, the participation plans generated by these heuristics may significantly deviate from optimal plans.

Alternatively, with our group-based heuristics we consider all users as a united group of participants. This allows for optimizing the assignment of the viewpoints among all users as a group; consequently, group-based heuristics can potentially generate near-optimal plans while maintaining high efficiency. In particular, group-based heuristics implement viewpoint assignment as a two-stage process. The main idea is to break the viewpoint assignment problem into multiple disjoint and smaller subproblems (at the first stage), where each subproblem takes a limited number of viewpoints as input and, therefore, can be solved efficiently (at the second stage). Accordingly, at the first stage group-based heuristics use various measures (e.g., proximity of the users to the viewpoints) to partition the set of viewpoints in V_S into a number of subsets, one subset per each user. At the second stage, similar to the individual-based heuristics, an orienteering heuristic is adopted to assign a subset of the viewpoints in each partition to the corresponding user of the partition.

4 GeoSIM: A Participatory Texture Documentation System

Figure 2 illustrates the client-server architecture of our PTD prototype system, dubbed GeoSIM. As depicted in the figure, the GeoSIM server consists of two engines, the planning engine and the texture mapping engine, which plan users participation and successively map the collected images, respectively. In addition to the viewpoint selection and viewpoint assignment modules which correspondingly implement our viewpoint selection and viewpoint assignment solutions described in Section 3, the planning engine includes a pre-imaging module that simulates the required images by imaging the corresponding area of the 3D environment model. The pre-imaged/simulated images are used to direct users in taking the required images properly.

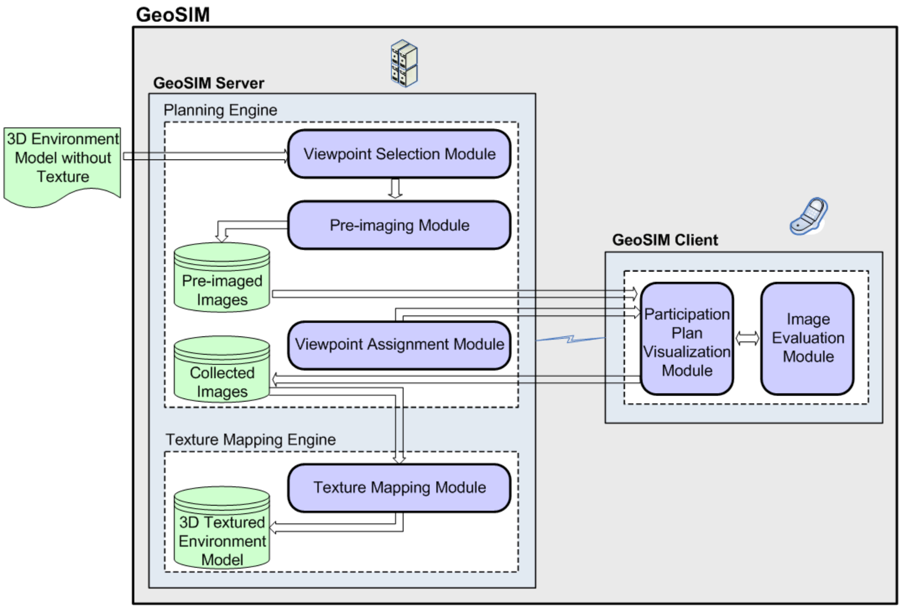


Figure 2: GeoSIM Architecture

The GeoSIM client (which is implemented as an Android application) comprises of two modules. The visualization module uses a Google Map^{TM} based interface to take user constraints, visualize the assigned user participation plan, and direct the user to take the required images. On the other hand, the image evaluation module considers various image features such as orientation, blurriness, and lighting to evaluate the quality of the images collected by the user. Accordingly, user is asked to re-take the images that are rejected by the evaluation module. Figure 3 shows the self-explanatory workflow of the participatory texture documentation process with GeoSIM. See [1] for more details about GeoSIM.

5 Related Work

Various commercial systems and research prototypes (e.g., Microsoft's Photosynth [2]) are developed that allow for texture mapping based on the images acquired by commodity cameras. In contrast, our focus is on effective planning to collect the images rather than merging the collected images for texture mapping.

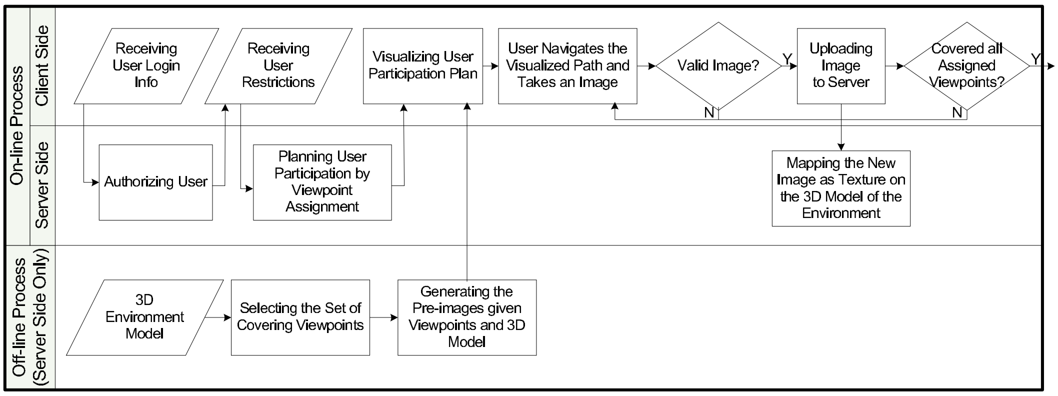


Figure 3: GeoSIM Workflow

6 Conclusion and Future Work

In this paper, we introduced the problem of geospatial planning for effective collection of texture information from urban environments. We also presented efficient heuristics that enable on-the-fly planning and described GeoSIM, the research prototype we have developed based on our solutions for participatory texture documentation. As part of our future work, we plan to extend GeoSIM to allow for documentation of data with other modalities, such as sound and temperature, and pollution.

Acknowledgement

This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0534761, and CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Geosim. http://infolab.usc.edu/projects/GeoSIM. Last Update: April 2010.

[2] Photosynth. http://photosynth.net/. Last Update: April 2010.

[3] I. Chao, B. Golden, and E. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.

[4] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *Proceedings of SODA*, January 2008.

[5] K. Chen and S. Har-Peled. The euclidean orienteering problem revisited. *SIAM Journal of Computing*, 38(1):385–397, 2008.

[6] H. Shirani-Mehr, F. Banaei-Kashani, and C. Shahabi. Efficient viewpoint assignment for urban texture documentation. In *Proceedings of ACMGIS*, November 2009.

[7] H. Shirani-Mehr, F. Banaei-Kashani, and C. Shahabi. Efficient viewpoint selection for urban texture documentation. In *Proceedings of GSN*, July 2009.

Spatio-Temporal Access Methods: Part 2 (2003 - 2010)*

Long-Van Nguyen-Dinh[1] Walid G. Aref[1] Mohamed F. Mokbel[2]

[1]Department of Computer Science, Purdue University, West Lafayette, IN 47907-1398

[2]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455

{lnguyend,aref}@cs.purdue.edu, mokbel@cs.umn.edu

Abstract

In spatio-temporal applications, moving objects detect their locations via location-aware devices and update their locations continuously to the server. With the ubiquity and massive numbers of moving objects, many spatio-temporal access methods are developed to process user queries efficiently. Spatio-temporal access methods are classified into four categories: (1) Indexing the past data, (2) Indexing the current data, (3) Indexing the future data, and (4) Indexing data at all points of time. This short survey is Part 2 of our previous work [28]. In Part 2, we give an overview and classification of spatio-temporal access methods that are published between the years 2003 and 2010.

1 Introduction

Spatio-temporal databases deal with moving objects that change their locations over time. Generally, moving objects report their locations obtained via location-aware devices to a spatio-temporal database server. The server can store all updates from the moving objects so that it is capable of answering queries about the past. Some applications need to know current locations of moving objects only. In this case, the server may only store the current status of the moving objects. To predict future positions of moving objects, the spatio-temporal database server may need to store additional information, e.g., the objects' velocities.

Many query types are supported by a spatio-temporal database server, e.g., range queries "*Find all objects that intersect a certain spatial range during a given time interval*", k-nearest neighbor queries "*Find k restaurants that are closest to a given moving point*", or trajectory queries "*Find the trajectory of a given object for the past hour*". These queries may execute on past, current, or future time data. A large number of spatio-temporal index structures has been proposed to support spatio-temporal queries efficiently.

In [28], we give a brief survey of spatio-temporal access methods that are published on or before 2003. This survey is Part 2 of [28], where we cover and classify spatio-temporal access methods published in the years 2003 to 2010. Since 2003, new spatial-temporal access methods have been developed that use entirely new approaches or that address weaknesses in existing approaches. Besides having separate indexing structures for past data, present data, or future data, some access methods have been proposed to deal with data at all points in time. Several spatio-temporal indexing methods are proposed for special environments, e.g., road networks or indoor networks. Figure 1 gives an overall overview of spatio-temporal access methods until 2010. Lines in the Figure indicate the evolutionary relationships among the spatio-temporal access methods.

The rest of this paper proceeds as follows. Sections 2, 3, and 4 survey spatio-temporal indexing methods for historical, current time, and future prediction data, respectively. Section 5 surveys spatio-temporal indexing methods for combined past, present, and future data. Section 6 gives concluding remarks.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*The authors acknowledge the support from the NSF under Grant No. IIS-0811954.

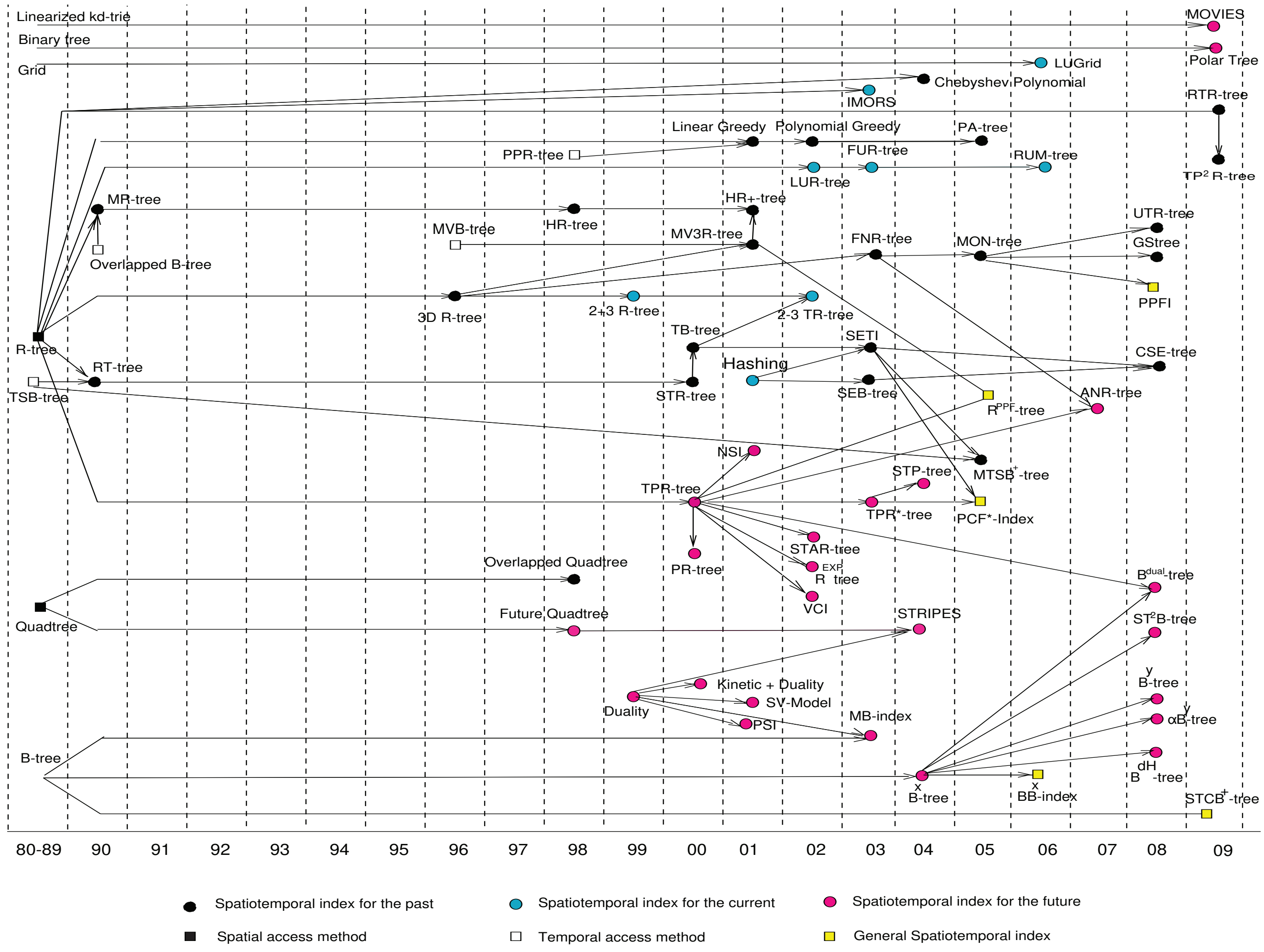


Figure 1: Survey of Spatiotemporal Access Methods.

2 Indexing the Past

In this section, we survey methods for indexing historical spatio-temporal data. We follow the same categorization as the one in Part 1 [28]. Since moving objects update their locations frequently, storing all historical data may not be feasible. Sampling helps reduce the size of stored historical data. Linear or non-linear interpolation techniques may be used between consecutive sampled data points to form trajectory paths. In contrast to sampling, objects may elect to update their locations only when they experience significant changes in location.

2.1 Three-dimensional Structures

In this category, time is modeled as the third dimension in addition to the two location dimensions (X,Y).

MTSB-tree [49]: The Multiple TSB-tree supports the historical and spatial range close-pair queries for moving objects. A range close-pair query finds pairs of objects that are closer than a given threshold distance during a given time interval and that both lay inside a given spatial range. Similar to SETI [4], in the MTSB-tree, space is divided into disjoint cells, each maintaining a temporal index for its objects' movements. Unlike SETI's R-tree-based [17] temporal index, the MTSB-tree uses a Time-Split B-tree (TSB-tree) [26] in each cell. Thus, all trajectory segments within the spatial and temporal range query are produced in increasing time order by merging the query results from the TSB-trees of all cells. Then, a stream of such trajectory segments is fed into a plane-sweep algorithm that sweeps both time and space to produce close-pair objects on the fly.

FNR-tree [16]: The Fixed Network R-Tree indexes objects moving in a two-dimensional (2D) fixed network. A fixed network is a set of connected line segments. The FNR-tree uses a 2D R-tree to index the static line segments of the network. Each leaf entry contains a line segment and a pointer to the root of a 1D R-tree that indexes the time intervals of objects' movements. Each time a moving object leaves a given line segment of the network, the FNR-tree searches the 2D R-tree to find the line segment and then inserts an interval (TEntrance, TExit) into the corresponding 1D R-tree. Every new entry is simply inserted in the rightmost leaf of the 1D R-tree. Thus, in the FNR-tree, the objects' movements are assumed not to begin or end in the middle of a line segment and their speeds or directions cannot be changed in the middle of a line segment.

MON-tree [10]: The MON-tree is an extension to the FNR-tree. In the MON-tree, the constrained network is modeled as a set of junctions (nodes) and routes (edges) in which routes are non-intersecting polylines. Unlike the FNR-tree where an object's position is updated only when the object leaves a line segment, in the MON-tree the object's position is associated with a real number between 0 and 1 that is the relative position of the object within a route. Similar to the FNR-tree, the MON-tree uses a top 2D R-tree to index the polylines' bounding boxes. For each polyline, a bottom 2D R-tree indexes the movements of the objects within the polyline. An object's movement is represented by a spatial range $(p1, p2)$ and a temporal interval $(t1, t2)$. The MON-tree uses a hash structure with the polyline identifier as the hash key to locate the polyline's corresponding bottom R-tree into which an object's movement is to be inserted.

2.2 Overlapping and Multi-version Structures

In this category, the temporal dimension is discriminated from the spatial dimensions in a variety of ways.

PA-tree [29]: The Parametric tree indexes historical trajectory data of moving objects by dividing the temporal domain into m disjoint time intervals. Each trajectory is split into a series of m line segments that correspond to the m disjoint time intervals and each segment is approximated with a single continuous Chebyshev polynomial. The PA-tree restricts the maximum deviation between the approximation and the original movement to make sure the approximation is conservative and tight. A two-level index indexes trajectory segments within each time interval. The first level is an R*-tree-like structure that indexes the two leading coefficients of the Chebyshev polynomial describing the object movement along each dimension and the maximum deviation errors. In the second level, higher-degree coefficients as well as the corresponding maximum deviations are stored for each trajectory segment. If the first level index is unable to determine whether the trajectory satisfies the query predicates, the additional coefficients in the second level index are used in the filtering step.

GStree [21]: The Graph Strip Tree indexes the current or past positions of moving objects in a constrained graph defined by edges and vertexes as in the MON-tree [10]. A strip tree [1] stores and models each edge that can be a list of line segments or curves by interpolating the edge linearly. The GStree is a balanced binary tree in which a leaf node is a minimum bounding area of an edge. These leaf nodes are merged two at a time to form internal nodes, such that the two merged nodes result in a minimum bounding area for their parent node. These internal nodes are merged in the same manner until the GStree reaches the root. The GStree assumes that all objects' positions are updated continuously in a constant update interval. It also assumes that there are $(m + 1)$ positions for each moving object defined on m equal intervals in the time domain $[0, T]$. Each leaf node of the GStree points to two data structures: the strip tree representing the edge stored in the leaf node, and an array of size m to store m interval trees of m time intervals. Each interval tree represents the moving objects' positions during the corresponding time interval on the corresponding edge. The strip trees and the GStree are stored in memory while the arrays are stored in disk. When new objects on an edge appear at a new time interval, a new interval tree containing these objects is created and is concatenated to the array.

2.3 Trajectory-oriented Access Methods

This category of access methods for historical spatio-temporal data deals with trajectory-oriented queries [35] that are either topological or navigational. Topological queries involve the topology of trajectories, e.g., check if an object enters or bypasses a given area. Navigational queries involve derived information, e.g., the maximum speed of an object during the last hour, or the orientation of an object (East, North, etc.).

CSE-tree [45]: The Compressive Start-End Tree has an idea similar to those of SETI [4] and the SEB-tree [40] to index the trajectories of moving objects. As in SETI, the space is partitioned into disjoint cells, and then a CSE-tree as a temporal index is maintained for each spatial cell. As in the SEB-tree, the CSE-tree stores a temporal interval as a 2D point. All segments indexed by a CSE-tree are divided into several groups according to the end time of the segment. A B+-tree-based End-Time index is built to index all groups. For each group, another B+-tree-based Start-Time index is organized to index all segments in that group. The CSE-tree is used in GPS data sharing applications, where people are interested in exchanging their GPS tracks. Thus, a track that consists of a sequence of track segments is inserted into the GPS data sharing system at the time a user uploads it, rather than at the time the GPS device updates a new location. Therefore, each Start-Time index has a different frequency update rate. When the update frequency of a Start-Time index goes below a threshold, the Start-Time index is transformed to a sorted dynamic array to utilize the storage.

Polar Tree [33]: The Polar Tree is an in-memory unbalanced binary tree to index object headings (orientations) of interest to a given focal point (site). Each site has a focal scope that represents its maximum range for detecting the objects moving in its vicinity. A focal scope is a circle of radius R centered at the fixed location of the site. An object's heading is represented as a counterclockwise angle w.r.t. the positive x-axis. By collecting objects' headings, the mode of progression can be observed. The Polar Tree divides the scope of a site into non-overlapping polar sectors corresponding to the nodes in the Polar Trees. The root node represents the entire scope of the site and has no entries. An internal node with a radius, say r, has exactly two children with radius $r/\sqrt(2)$ and the central angle w of the internal node is bisected into equal parts for its children. When a leaf node overflows, it becomes an internal node with two new children. Each object's heading is inserted at suitable internal nodes or leaves of the Polar Tree built for a specific focal point. Therefore, the Polar Tree is used to monitor objects' orientations and detect if many objects get closer to or move away from a site.

Chebyshev Polynomial [3]: A d-dimensional spatio-temporal trajectory is projected to its d dimensions to have d 1D series that are each approximated by a Chebyshev polynomial [27] and is represented by a vector of coefficients. All coefficient vectors are combined to form one vector or one single multi-dimensional point representing the trajectory. An arbitrary multidimensional access method indexes the points of all trajectories. It assumes that all trajectories are sampled at the same set of timestamps. Thus, the polynomial approximations for all trajectories have the same degree and the time dimension can be ignored. The Lower Bounding Lemma states that the Euclidean distance between two trajectories is lower bounded by the Euclidean distance between the two vectors of Chebyshev coefficients weighted by a factor. Based on the lemma, a query to find all trajectories within a range from a given trajectory or a kNN query of a specified trajectory can be answered using the index.

RTR-tree and TP^{2}R-tree [19]: The indoor movement is more constrained than outdoor Euclidean movement and is characterized by entities (doors, rooms, hallways, etc.) and topology predicates (enter, leave, cross, etc.). Location data is collected by RFID readers that are deployed at fixed locations in the indoor space to locate moving objects. The RTR-tree and TP^{2}R-tree [19] are two R-tree-based indexes for trajectories of objects moving in symbolic indoor space. The RTR-tree is a 2D R-tree on the Reader-Time space that organizes a trajectory as a list of line segments. To answer a range query, the Euclidean range space is transformed into sets of RFID readers and then the RTR-tree is searched for the corresponding readers. The TP^{2}R-tree uses the same Reader-Time space, but transforms a trajectory to a set of points, augmented with a temporal parameter. This way, the TP^{2}R-tree achieves better node organization.

3 Indexing the Current Positions (NOW)

The notion of the "current" or NOW in database systems is challenging (e.g., see [9]). In this section, we give an overview of spatio-temporal access methods that answer queries on the NOW status of moving objects.

LUGrid [47]: The Lazy-Update Grid-based index adapts the grid file [30] to exploit lazy insertion and deletion to handle the frequent updates to the locations of continuously moving objects. Incoming updates are grouped based on the to-be-updated disk-page, are stored in a memory grid, and then are lazily flushed into disk in batch. Thus, multiple updates are reduced to only a single disk update. In lazy deletion, obsolete entries remain temporarily in disk rather than being immediately deleted. The LUGrid uses an in-memory memo structure to keep track of current object ids. This way, the obsolete entries can be referred from the current entries in the memo and are lazily removed when their disk pages are accessed.

RUM-tree [46, 39]: The R-tree with update memo deals with the frequent updates of moving objects. Instead of deleting the object from the old location and reinserting it into the new location, upon a location update, we just insert the object in the new location. This results in multiple obsolete (old) locations of an object. As in the LUGrid [47], the RUM-tree uses an in-memory memo to track the obsolete entries to avoid purging old entries immediately during an update process. Upon touching a disk page, say p, e.g., during query processing, and before reporting an object, say o, as output, o's version in p is checked against the memo to make sure that o's version is current. A garbage cleaner (GC) inside the RUM-tree removes the obsolete entries lazily and makes sure that the size of the in-memory memo is bounded. Various mechanisms are used by GC to remove obsolete entries including a vacuum cleaner approach that regularly sweeps the space with some frequency and a clean-upon-touch approach that cleans a leaf node whenever it is fetched during an insert or update operation.

IMORS [20]: Indexing Moving Objects on Road Sectors indexes the current positions of objects with high update frequencies moving on a fixed road network. IMORS uses an R*-tree to index road sectors (non-intersecting segments of a road, each bounded by two intersection points). Each road sector entry on leaf nodes of the R*-tree points to a road sector block (BRS) that contains the identifiers of the moving objects on the road sector. In addition, data blocks (Bdata) store data records of the velocities and coordinates of moving objects. There are bi-directional pointers between each record in Bdata and a corresponding moving object in a BRS. To insert a new moving object into IMORS, one data record with the object's coordinates and other attributes is inserted into a Bdata and the object identification (oid) is inserted into a BRS corresponding to a road sector where the new object is contained. To update a new position of a moving object, first, the record of the object from the Bdata is retrieved by its oid and then is updated. If the object is still on the same BRS, no more operations are needed. Otherwise, the oid entry in the old BRS found directly by the pointer from the data record is deleted and is inserted into the new BRS. The road sector can be found by searching the R*-tree.

4 Indexing the Future Positions

To index the future positions of moving objects, access methods predict the future locations of the objects in a variety of ways, e.g., using the objects' reference locations and their velocities.

4.1 The Original Space-time Space

Here, the predicted positions of moving objects are calculated and plotted in the original spatio-temporal space.

MOVIES [12]: The main-memory MOVing objects Indexing using frEquent Snapshots supports predictive queries on moving objects. Instead of creating only one index and modifying it according to incoming updates, MOVIES constructs an index w.r.t. the most updated data of moving objects and uses that index for a short period of time and then throws it away when a new index is constructed. Thus, MOVIES provides a read-optimized index. Query results may not consider the most recent updates if the updates arrive before a new index is scheduled for construction. MOVIES uses linearized kd-tries [31, 44] to index moving objects' locations. It uses Predictive (PI) or Non-Predictive Indexing (NPI). In PI, an object's predicted position is translated immediately to be indexed in MOVIES when an update arrives using the moving object's linear function. In contrast, NPI does not compute predicted positions at indexing time but rather at the query processing time.

4.2 Transformation Methods

Here, the time-space domain is transformed to a space so that it is easier to represent and query data in the future.

4.2.1 Duality transformation

STRIPES [32]: The Scalable Trajectory Index for Predicted Positions in Moving Object Databases models the movements of an object as a linear function in a d-dimensional space. It transforms the predicted positions of a moving object in the d-space into points in a 2d dual space (storing the coordinates and the velocities of the moving objects). Two distinct indexes are maintained. The first and second indexes cover the time intervals from 0 to L and from L to $2L$, respectively. During every time period L, objects update their locations and velocities by being inserted into the second index and being deleted from the first index. As time elapses, the first index becomes empty and the second index is populated. Each STRIPES index is a disk-based bucket PR quadtree [37], where each dual plane (Vi, Pi) is partitioned equally into four quadrants, with Vi and Pi being the velocity and coordinate values at Dimension i in the original d-space. Because STRIPES may lead to low page utilization, it stores leaf nodes with occupancy less than 50% in half a page and leaf nodes with over 50% occupancy in a full page. Insertion, deletion, and update are as in the PR quadtree. Update tuples contain both the old and new locations, hence an update is a delete of the old location followed by an insert of the new one.

MB-index [14]: The MB-index transforms the objects in d-space to a 2d-space that corresponds to the objects' locations and velocities. One dimension is kept fixed and for each other dimension, (d-1) dimensional hyperplanes that are perpendicular to that dimension partition the space so that the points distribute uniformly among partitions. For each partition, a B-tree indexes the points ordered by the value of the fixed dimension. A range query is transformed into a 2d-polyhedron. Then, each hyperplane of the polyhedron is intersected with each partition to find the point candidates that are then checked for inclusion inside the query polyhedron.

4.2.2 Space-filling-curve (SFC) Transformation

B^x-tree [18]: A B^x-tree extends the B$^+$-tree to index moving objects. Object locations are transformed to 1d values using an SFC that preserves location proximity. The B^x-tree partitions the time axis into equal intervals corresponding to the maximum duration between two updates of any object location. Each interval is then sub-partitioned into equal phases. When one moving object updates its location, the B^x-tree computes the predicted location of the object at the end timestamp of the next phase using the moving object's linear time function. Then, the B^x-tree uses an SFC to convert this predicted location to a 1d-value. This value and the partition information are concatenated and indexed by the B^x-tree. A range query is answered by an iterative query enlargement algorithm to explore appropriate partitions. Deletion is the same as in the B$^+$-tree. The positional information for the object at its insertion and the last insertion time are assumed to be known. Unlike update in the B$^+$-tree, an object in the B^x-tree is only updated when its linear moving function changes. Upon update, the B^x-tree deletes the object out of its tree partition and then reinserts the updated value into the most recent partition. Similar to STRIPES [32], the B^x-tree keeps two partitions at a time.

B^y-tree and α B^y-tree [6]: These two indexes extend the B^x-tree [18] by using separate update frequencies for each moving object. Thus, the B^y-tree and the α B^y-tree work well in the environments with hightly variable update periods. The α B^y-tree uses the parameter α to balance the performance of updates and queries. Instead of updating values every phase (as in the B^x-tree), the α B^y-tree updates values after every α phases.

ST^{2}B-tree [8]: The ST^{2}B-tree indexes the future positions of moving objects in exactly the same way as that of the B^x-tree. However, it improves on the B^x-tree by adapting to data skewness in space and time. The ST^{2}B-tree partitions the space into disjoint Voronoi cells using a set of reference points identified dynamically. Based on the data density within a Voronoi region, each region has an individual grid. A moving object is assigned to one grid cell within a Voronoi region. A tree partition in the ST^{2}B-tree grows and shrinks one after the other over time. The Voronoi regions and their corresponding grids are adjusted given the new data distribution.

B^{dual}-tree [48]: The B^{dual}-tree builds on the B^x-tree. Unlike the B^x-tree that indexes only objects' locations, the B^{dual}-tree captures both d-dimensional locations and velocities in a dual 2d-space. The dual space is partitioned along all dimensions into cells. Then, an SFC transforms the 2d-values in the dual space into 1d-values that are indexed in a B$^+$-tree. Any cell in the partitioning space can be regarded as a d-dimensional moving rectangle (MOR) that captures the locations and velocities of all objects inside it similar to the time parameterized bounding rectangle (TPBR) in the TPR-tree [36]. Each internal entry in the B$^+$-tree maintains a set of MORs, indicating the spatial region and range of velocity covered by the sub-tree of the entry. Unlike the B^x-tree where spatiotemporal search is reduced to several 1D range queries on the B$^+$-tree, the B^{dual}-tree uses

R-tree-like query algorithms as in the TPR-tree. A range query searches the sub-tree of an internal entry only if any of its MORs intersect the query region. Insertion, update and deletion of a moving object in the B^{dual}-tree are similar to those of the B^x-tree.

B^{dH}-tree [38]: The B^{dH}-tree builds on the B^x-tree. Instead of using a B^+-tree, the B^{dH}-Tree uses a B^{link}-tree [41] that is a B^+-tree whose internal nodes at the same level are linked. Unlike the B^x-tree, the B^{dH}-tree transforms data using a dynamic Hilbert SFC whose order can be automatically adjusted according to data distribution. Thus, sub-regions with different data distributions in space have different Hilbert SFC resolutions. Merging subregions lowers their SFC order while splitting a subregion causes the divided subregions to have a higher order. Insertion, deletion and update are the same as those of the B^x-tree.

To sum up, dual transformation techniques suffer from three main drawbacks: (1) The *dual* space does not capture all the information in the *primal* space. (2) There is no guarantee that objects near each other in the *primal* space will still be near each other in the *dual* space. (3) Rectangular range queries in the *primal* space are always transformed into polygonal range queries in the *dual* space, which calls for more complex algorithms.

4.3 Time-parameterized Access Methods

STP-tree [42]: The Spatio-temporal Prediction Tree generalizes the TPR [36] and TPR*-trees [43] by arbitrary polynomial moving functions to index future positions of moving objects. It assumes that each moving object, say o, maintains the most-recent locations and is able to continuously revise its individual motion function, say $o(t)$. While different objects can follow different motion functions, the server indexes the same motion type for all objects. The motion type is a polynomial function of any degree D that approximates the moving function $o(t)$ in the d-dimensional space. The precise motion functions $o(t)$ are used only when the server cannot determine if a candidate qualifies predictive range queries. In contrast to the TPR*-tree, the parameterized MBR in the STP-tree is represented by two polynomials over time starting at two opposite corners of a d-dimensional rectangle enclosing the locations of the children through the maximum timestamp of all children. Insertion, update, and deletion in the STP-tree are similar to those in the TPR*-tree. The nodes' bounding polynomial matrices guide the search for objects, given the objects' polynomials.

ANR-tree [5]: The Adaptive Network R-tree indexes the future trajectories of objects in a constrained network. As in the FNR-tree [16], the ANR-tree is a two-level index. At the top, a 2D R-tree indexes the network. At the bottom level, another 1D R-tree is used to index adaptive units. An adaptive unit extends a 1D MBR of the TPR-tree by grouping adjacent objects with the same direction and similar speed. The predicted movement of an object is not a linear function, but is bounded by the fastest and slowest movements of the object.

5 Access Methods for Past, Present, and Future Spatio-temporal Data

R^{PPF}-tree [34]: The R^{PPF}-tree (Past, Present, and Future) indexes positions of moving objects at all points in time. The past positions of an object between two consecutive samples are linearly interpolated and the future positions are computed via a linear function from the last sample. The R^{PPF}-tree applies partial persistence to the TPR-tree to captures the past positions. Leaf and non-leaf entries of the R^{PPF}-tree include a time interval of validity - [insertion time, deletion time]. When a node, say x, is split at time t, entries in x alive at t are copied to a new node, say y and their timestamps are set to [t, ?) (i.e., their deletion times are unidentified). While a time-parameterized bounding rectangle (TPBR) of the TPR-tree is valid from the current time, the structure of a TPBR in the R^{PPF}-tree is valid from its insertion time. The *straightforward*, *optimized*, and *double* TPBRs are studied. In the straightforward approach, the bounding rectangle is the integral of the TPBR from its insertion time to infinity. In the optimized TPBR, the bounding rectangle is the integral of the TPBR from its insertion time to (current time + H time units in the future that can be efficiently queried). The straightforward and optimized TPBRs cannot be tightened since these rectangles start from their insertion times. The double TPBR allow tightening by having two components: a tail MBR and a head MBR. The tail MBR starts at the time of the last update and extends to infinity and thus is a regular TPBR of the TPR-tree. The head MBR bounds the finite historical trajectories from the insertion time to the last update. Querying is similar to the regular TPR-tree search with the exception of redefining the intersection function to accommodate for the double TPBR.

PCFI$^+$-index [25]: The Past-Current-Future+-Index builds on SETI [4] and TPR*-tree [43]. As in SETI, space is divided into non-overlapping cells. For each cell, an in-memory TPR*-tree indexes the current positions and velocities of objects. Current data records are organized as a main-memory hash index, hence allowing efficient access to current positions. To index the objects' past trajectories, the PCFI$^+$-index uses a sparse on-disk R*-tree to index the lifetimes of historical data that only contains the segments from one cell. Insertion, update, and deletion are similar to those of SETI and TPR*-tree. Upon update, if the new location resides inside the same partition, a new segment is inserted into the historical data file; the TPR*-tree updates the new location for the object. Otherwise, a split occurs and two segments are inserted into the historical data file at different pages; the corresponding entry in the old TPR*-tree is removed and is inserted into another TPR*-tree. If the insertion of a segment overflows a page, the corresponding R*-tree entry is updated to set its end time.

BB^x-index [22]: The BB^x-index uses the B^x-tree techniques to support the present and future. To index the past, the BB^x-index keeps multiple tree versions. Each tree entry has the form $< x\_rep, tstart, tend, pointer >$, where x_rep is the transformed 1D object location value using an SFC, $tstart$ and $tend$ are the insert and update times of the object, respectively. Each tree corresponds to a timestamp signature being the end timestamp of a phase when the tree is built and a lifespan being the minimum start time and the maximum end time of all the entries in that tree. Unlike the B^x-tree that concatenates the timestamp signature and the 1D transformed value, the BB^x-index maintains a separate tree for each timestamp signature, and models the moving objects from the past to the future. Insertion is the same as in the B^x-tree. Instead of deleting an object, update sets the end time of the object to the current time, followed by an insertion of the updated object into the latest tree.

UTR-tree [11]: The Uncertain Trajectory R-tree extends the MON-tree [10] with uncertainty within a constrained network. A top R-tree indexes directed atomic route sections (ARSs) that connect two junctions of a route and do not contain any other junctions from which moving objects can exit the traffic flow. Leaf entries that corresponds to the ARSs of the same route have a pointer to a bottom R-tree of the route. The bottom R-trees of the UTR-tree keep the latest locations and trajectories of moving objects, and thus support historical, present, and near-future queries. Between two exact locations of the moving object at two consecutive update times, the UTR-tree derives the uncertain trajectory of the object and indexes it in the corresponding bottom R-tree.

STCB$^+$-tree [23]: The Spatio-temporal Compressed B$^+$-tree uses multiple Compressed B$^+$-trees [24] (a CB$^+$-tree is a B$^+$-tree with at least 75%-full leaf nodes) to index trajectories of moving objects. One CB$^+$-tree, termed the TCB$^+$-tree, indexes the temporal dimension and for each spatial dimension, one CB$^+$-tree, termed the SCB$^+$-tree, indexes the spatial attributes of moving objects in that dimension. The temporal and spatial coordinates of endpoints of all trajectory segments are indexed by the CB$^+$-trees. Insertion and deletion in the CB$^+$-tree are the same as in the B$^+$-tree. In the TCB$^+$-tree, each interval identified by two consecutive key values stores a bucket including the identifiers of the objects moving in that interval. The generation of new temporal data is totally ordered. Thus, insertions into the TCB$^+$-tree append new entries to the rightmost leaf. The SCB$^+$-tree distinguishes the onward, backward, and still motions of a trajectory according to the departure and arrival positions of the trajectory. Similar to the TCB$^+$-tree, for each interval identified by two consecutive key values, the SCB$^+$-tree keeps a bucket including the identifiers and locations of objects moving within the interval. To answer a spatio-temporal query, the TCB$^+$- and the SCB$^+$-trees are searched for the temporal and spatial output, respectively; the final output is the object identifiers common in all the outputs.

PPFI [15]: The PPFI uses a 2D R*-tree to index a road network. Each road sector has a 1D R*-tree (Rs) that index the time interval for the past trajectory of objects moving along the sector. Pointers link the leaf entries created by the consecutive updates of the same object. These leaf entries can be in the same Rs or in different Rs's. A hash structure describes the recent state of moving objects and predicts their near-future positions using the objects' linear moving functions. To insert a new moving object into the PPFI, two entries for the object are inserted into (1) the hash and (2) the leaf node of the corresponding Rs. These two entries point to each other. Using this hybrid data structure, the PPFI can support trajectory-based and predictive queries.

6 Conclusion

This short survey is a continuation of [28]. It classifies and overviews existing spatio-temporal access methods for the years 2003-2010. Although not the scope of this survey, it is important to highlight the works in [2, 7, 13] that provide spatio-temporal data generators and benchmarks to generate datasets for both general or road network environments and facilitate the evaluation of spatio-temporal access methods under various conditions.

References

[1] D. H. Ballard. Strip trees: a hierarchical representation for curves. *Commun. ACM*, 24(5):310–321, 1981.

[2] T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.

[3] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. *SIGMOD*, pp. 599–610, 2004.

[4] V. Chakka, A. Everspaugh, and J. Patel. Indexing large trajectory data sets with SETI. In *CIDR*, 2003.

[5] J.-D. Chen and X.-F. Meng. Indexing future trajectories of moving objects in a constrained network. *J. Comput. Sci. Technol.*, 22(2):245–251, 2007.

[6] N. Chen, L.-D. Shou, G. Chen, and J.-X. Dong. Adaptive indexing of moving objects with highly variable update frequencies. *J. Comput. Sci. Technol.*, 23(6):998–1014, 2008.

[7] S. Chen, C. Jensen, and D. Lin. A benchmark for evaluating moving object indexes. *PVLDB*, 1(2):1574–1585, 2008.

[8] S. Chen, B. Ooi, K. Tan, and M. Nascimento. ST^2B-tree: A self-tunable spatio-temporal B^+-tree index for moving objects. In *SIGMOD*, pages 29–42, 2008.

[9] J. Clifford, C. Dyreson, T. Isakowitz, C. Jensen, and R. Snodgrass. On the semantics of “Now” in databases. *ACM TODS*, 22(2), 1997.

[10] V. T. De Almeida and R. H. Güting. Indexing the trajectories of moving objects in networks. *Geoinformatica*, 9(1):33–60, 2005.

[11] Z. Ding. UTR-tree: An index structure for the full uncertain trajectories of network-constrained moving objects. In *MDM*, pages 33–40, 2008.

[12] J. Dittrich, L. Blunschi, and M. A. Vaz Salles. Indexing moving objects using short-lived throwaway indexes. In *SSTD*, pages 189–207, 2009.

[13] C. Düntgen, T. Behr, and R. H. Güting. BerlinMOD: A benchmark for moving object databases. *The VLDB Journal*, 18(6):1335–1368, 2009.

[14] K. M. Elbassioni, A. Elmasry, and I. Kamel. An efficient indexing scheme for multi-dimensional moving objects. In *ICDT*, pages 422–436, 2003.

[15] Y. Fang, J. Cao, Y. Peng, and L. Wang. Indexing the past, present and future positions of moving objects on fixed networks. In *CSSE (4)*, pages 524–527, 2008.

[16] E. Frentzos. Indexing objects moving on fixed networks. In *SSTD*, pages 289–305, 2003.

[17] A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, June 1984.

[18] C. Jensen, D. Lin, and B. Ooi. Query and update efficient B^+-tree based indexing of moving objects. In *VLDB*, 2004.

[19] C. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *SSTD*, 2009.

[20] K.-S. Kim, S.-W. Kim, T.-W. Kim, and K.-J. Li. Fast indexing and updating method for moving objects on road networks. *Web Information Systems Engineering Workshops*, 0:34–42, 2003.

[21] T. T. T. Le and B. G. Nickerson. Efficient search of moving objects on a planar graph. In *GIS*, pages 1–4, 2008.

[22] D. Lin, C. Jensen, B. Ooi, and S. Šaltenis. Efficient indexing of the historical, present, and future positions of moving objects. In *MDM*, pages 59–66, 2005.

[23] H. Lin. Indexing the trajectories of moving objects. In *Intl. MultiConf. of Engrs. and Computer Scientists*, pages 732–737, 2009.

[24] H. Lin, R. Chen, and S. Chen. Enhancement of data aggregation using a novel point access method. *WSEAS Transactions on Computers*, 7(12):2001–2010, 2008.

[25] Z.-H. Liu, X.-L. Liu, J.-W. Ge, and H.-Y. Bae. Indexing large moving objects from past to future with PCFI+-index. In *COMAD*, pages 131–137, 2005.

[26] D. Lomet and B. Salzberg. Access methods for multiversion data. In *SIGMOD*, pages 315–324, May 1989.

[27] J. Mason and D. C. Handscomb. *Chebyshev polynomials*. Chapman and Hall, 2003.

[28] M. Mokbel, T. Ghanem, and W. G. Aref. Spatio-temporal access methods. *IEEE Data Eng. Bull.*, 26(2):40–49, 2003.

[29] J. Ni and C. V. Ravishankar. PA-tree: A parametric indexing scheme for spatio-temporal trajectories. In *SSTD*, 2005.

[30] J. Nievergelt, H. Hinterberger, and K. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *TODS*, 9:38–71, 1984.

[31] J. Orenstein and T. Merrett. A class of data structures for associative searching. In *PODS*, pages 181–190, 1984.

[32] J. Patel, Y. Chen, and V. Chakka. STRIPES: An efficient index for predicted trajectories. *SIGMOD*, 2004.

[33] K. Patroumpas and T. Sellis. Monitoring orientation of moving objects around focal points. *SSTD*, pp 228-246, 2009.

[34] M. Pelanis, S. Šaltenis, and C. Jensen. Indexing the past, present, and anticipated future positions of moving objects. *TODS*, 31(1):255–298, 2006.

[35] D. Pfoser, C. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *VLDB*, pages 395–406, 2000.

[36] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD*, pages 331–342, 2000.

[37] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

[38] D. Seo, S. Song, Y. Park, J. Yoo, and M. Kim. B^{dH}-tree: A B^{+}-tree based indexing method for very frequent updates of moving objects. *Intl. Symp. on Computer Sc. and its Applications*, 0:314–319, 2008.

[39] Y. N. Silva, X. Xiong, and W. G. Aref. The RUM-tree: Supporting frequent updates in R-trees using memos. *VLDB J.*, 18(3):719–738, 2009.

[40] Z. Song and N. Roussopoulos. SEB-tree: An approach to index continuously moving objects. In *MDM*, 2003.

[41] V. Srinivasan and M. Carey. Performance of B-tree concurrency control algorithms. *SIGMOD Rec.*, 20(2):416–425, 1991.

[42] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD*, pages 611–622, 2004.

[43] Y. Tao, D. Papadias, and J. Sun. The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In *VLDB*, 2003.

[44] H. Tropf and H. Herzog. Multidimensional range search in dynamically balanced trees. *Angewandte Informatik*, 23(2):71–77, 1981.

[45] L. Wang, Y. Zheng, X. Xie, and W.-Y. Ma. A flexible spatio-temporal indexing scheme for large-scale GPS track retrieval. In *MDM*, pages 1–8, 2008.

[46] X. Xiong and W. G. Aref. R-trees with update memos. In *ICDE*, page 22, 2006.

[47] X. Xiong, M. Mokbel, and W. G. Aref. LUGrid: Update-tolerant grid-based indexing for moving objects. In *MDM*, page 13, 2006.

[48] M. Yiu, Y. Tao, and N. Mamoulis. The B^{dual}-tree: Indexing moving objects by space filling curves in the dual space. *VLDB J.*, 17(3):379–400, 2008.

[49] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios. Close pair queries in moving object databases. In *GIS*, pages 2–11, 2005.

SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations

Ralf Hartmut Güting Thomas Behr
Christian Düntgen
Faculty of Mathematics and Computer Science
University of Hagen, 58084 Hagen, Germany
{rhg, thomas.behr, christian.duentgen}@fernuni-hagen.de

Abstract

Databases supporting time dependent and continuously changing geometries, called moving objects databases*, have been studied for about 15 years. The field has been flourishing and there exist many hundreds, more likely thousands, of publications. However, very few of these results have made it into systems (research prototypes or commercial) and are available for practical use today.*

It is not that the publications are purely theoretical. In most cases data structures and algorithms have been proposed, implemented, and experimentally evaluated. However, whereas there exists a well established infrastructure for publishing research papers through journals and conferences, no such facilities exist for the publication of the related prototypical implementations. Hence implementations are just done for experiments in the paper and then usually abandoned. This is highly unfortunate even for research, as future proposals of improved algorithms most often have to reimplement the previous techniques they need to compare to.

In this paper we describe an infrastructure for research in moving objects databases that addresses some of these problems. Essentially it allows researchers to implement their new techniques within a system context and to make them available for practical use to all readers of their papers and users of the system. The infrastructure consists of SECONDO*, an extensible database system into which a lot of moving object technology has been built already. It offers BerlinMOD, a benchmark to generate large sets of realistic moving object trajectories together with a comprehensive set of queries. Finally, it offers* SECONDO *Plugins as a facility to publish new research implementations that anyone can merge with a standard* SECONDO *distribution to have them run in a complete system context.*

1 Introduction

Moving objects databases allow one to represent and query in a database the time dependent location of mobile entities such as vehicles or animals, either online for current movement, or offline storing large sets of trajectories, or histories of movement. This has been a very active field of research since about the mid-nineties. There exist probably several thousand related publications. A large fraction of them addresses practical issues

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

such as indexing and query processing algorithms and the respective paper describes an implementation and an experimental evaluation. However, only very few of these implementations have ever been integrated into some larger system environment, and only a few of them are still available today for practical use or comparison with other solutions. The basic reason is that the current methodology and infrastructure for experimental research in databases is geared to publishing papers, not software. As a consequence, the field has grown much more in terms of papers than of systems.

This is unfortunate as the transfer of research results to practical applications is quite difficult. Reimplementing published solutions may require several months of work. Combining many such solutions into an application will often be not feasible. It is also unfortunate for research itself as the quality of research deteriorates (as explained below) and a lot of unnecessary double work arises.

The current methodology can roughly be described as follows. A new data structure or algorithm is proposed, say, a new type of index structure, or a query processing algorithm. The authors describe their proposal and implement it. To prove the new proposal is worth publishing, they have to provide an experimental evaluation which needs to include a comparison with the strongest competing proposals from the literature. Unfortunately in most cases the competing algorithms need to be reimplemented, because the original implementations of their authors are not available or suitable for comparison. Hence the authors take the effort to also reimplement the competitors, perform their experiments and report them in the paper. Assuming the paper is accepted, this is the end of the story. The implemented software is abandoned. Anyway, it is only suitable to be used in a very specialized system context for performing experiments and doing measurements. There is no way to use it in a practical system or in real applications.

Original implementations of competing proposals may be available on request from the authors in a few lucky cases, but most often they are not. There are many reasons: The Ph.D. student who did it, left. The software was written some years ago and not maintained. It was buggy in the first place and barely able to execute the experiments of the original paper (omitting those that did not work). It was undocumented. It was written in a different programming language. And so forth.

That these algorithms need to be reimplemented by the authors of another proposal is bad for several reasons. First, it is a waste of resources. Second, there is a great danger that in the reimplementation errors are made. Even with the best effort, it is easily possible that subtle points in the descriptions in the respective papers have been misunderstood. Possibly some issues have not even been described clearly or at all. Third, the authors of the new proposal are of course interested in demonstrating that their new algorithm is better than the competitors. It exercises a lot of discipline in them to make sure that within the competing implementations everywhere the most efficient technique is used and minor details, that however might severely deteriorate performance, are treated right.

The lack of the software being published with the paper also has a negative impact on the scientific quality of the publication. Authors design certain experiments with certain data sets, varying some parameters. Although referees try to make sure that this has been done carefully, in many cases questions remain. How would this algorithm behave for this other parameter combination? What were the exact properties of the data set? Could they have had a special impact on this algorithm? If the competing algorithms were available with the publication and could easily be run with other parameters or data sets, all such questions could be clarified. Definitely results would be more reliable. Moreover, even years after the publication, issues could be reexamined.

The research community is aware of some of these issues. For example, there is a trend to encourage experimental repeatability, as shown at the last SIGMOD conferences. VLDB has an "Experiments and Analyses Track" that aims at providing a prestigious forum for careful experimental investigation of known techniques.

In this paper, we propose an infrastructure that allows authors to publish their research implementations together with the papers in such a way that readers of the paper can directly run the software, repeat the experiments, and even extend the experiments using other parameters and data sets. The infrastructure is especially attractive for research implementations for moving objects databases. This is because large parts of the MOD data models of [12, 13] have been implemented, benchmark data and queries [7] are available, and spatial as well as moving objects can be visualized and animated.

The infrastructure consists of the SECONDO system, a DBMS prototype that has been designed as an extensible system from the beginning. SECONDO is in particular extensible by so-called algebra modules. Furthermore, it includes the plugin facility which allows one to describe and pack extensions.

The basic idea is that the author A of a paper wraps her implementation as a SECONDO algebra module, packs it into a plugin, and publishes it on her web site. The reader can get a SECONDO system from the standard distribution, get the plugin from the author's web site, and merge them by a simple command. The reader can then try algorithms in SECONDO and redo and extend experiments.

A future researcher who wishes to propose an improvement to $A's$ algorithm can implement the new algorithm in SECONDO and compare to $A's$ original implementation by just running it.

In the following sections we describe in a bit more detail the architecture of SECONDO, the moving objects implementations that are available, the BerlinMOD benchmark, and the plugin facility.

2 SECONDO

Research implementations for papers, as discussed in the previous section, might be integrated into various DBMS environments. For moving objects related implementations, PostGIS or OracleSpatial might be interesting candidates. In this section we argue that SECONDO is a particularly suitable environment. SECONDO is a prototype DBMS developed at University of Hagen since about 1995. It runs on Windows, Linux, and MacOS X platforms and is freely available open source software [4]. The main design goals were a clean extensible architecture and support for spatial and spatio-temporal applications. In the sequel we address the following questions:

- Why is SECONDO suitable for publishing research extensions?
- Why is SECONDO suitable for publishing implementations related to moving objects?

We also provide an example of an extension and explain how it benefits from the environment.

2.1 Extensible Architecture

SECONDOs architecture is shown in Figure 1. It consists of three major components: the kernel, the optimizer, and the GUI. The kernel does not implement a fixed data model but is open for implementation of a wide

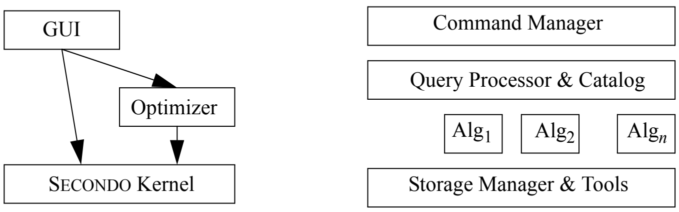


Figure 1: SECONDO components (left), architecture of kernel system (right)

variety of DBMS data models. The kernel is extensible by algebra modules. In fact, the entire implementation of a particular data model is done within algebra modules. An algebra module generally offers a set of type constructors and a set of operators. A type constructor is a (parameterized) data type and is defined via a signature.

$\underline{int}$, $\underline{real}$, $\underline{bool}$, $\underline{string}$:		$\rightarrow$ DATA
$\underline{tuple}$:	(IDENT $\times$ DATA)+	$\rightarrow$ TUPLE
$\underline{rel}$:	TUPLE	$\rightarrow$ REL

For example, the above signatures define type constructors for the relational model. Here $\underline{rel}$ is a type constructor applicable to all types in the kind TUPLE, returning a type in kind REL. The terms generated by such signatures define the available data types. Over the data types, operations can be defined, for example

$$
\begin{array}{lll}
+: & \underline{int} \times \underline{int} & \rightarrow \underline{int} \\
feed: & \underline{rel}(tuple) & \rightarrow \underline{stream}(tuple) \\
filter: & \underline{stream}(tuple) \times (tuple \rightarrow \underline{bool}) & \rightarrow \underline{stream}(tuple) \\
consume: & \underline{stream}(tuple) & \rightarrow \underline{rel}(tuple)
\end{array}
$$

An algebra module implements its type constructors and operators. Essentially for a type constructor a data structure and for an operator an evaluation function need to be provided. Some more support functions must be written, e.g. type checking functions for type constructors and for operators.

Algebra modules encapsulate everything needed to implement a DBMS data model, hence there are algebras in SECONDO for basic data types, for relations and tuples including operations such as hashjoin, for B-Trees and R-trees with their access operations, for spatial and spatio-temporal data types, and many more. There are also algebras beyond the scope of a relational model such as for nested relations or networks.

The kernel is an engine to evaluate terms over the existing objects and operations. For example, it evaluates expressions:

```
query 3 + 5
query Trains feed filter[.Trip present sixthirty]
  filter[length(trajectory(.Trip)) > 2000.0] count
```

Here *Trains* is a relation containing trajectories represented in an attribute *Trip* (of type $\underline{mpoint}$ = moving point in SECONDO). Syntax for operations can be freely chosen and it is often practical to use postfix notation for query processing operations. For example, the first argument to *filter* is *Trains feed*. Stream processing is built into the engine. The commands and queries processed directly by the kernel are called the *executable language*. The kernel is written in C++ and uses BerkeleyDB as the underlying storage manager.

The optimizer is not as data model independent as the kernel. It assumes an object relational model and supports an SQL-like language. It maps SQL to the executable language shown above. The optimizer is extensible by registering types and operations of the executable level, by translation rules, and cost functions. It allows for extension by new index types, providing concepts to distinguish between logical and physical indexes (a physical index is a particular index structure available in the kernel, a logical index is a strategy to use it, which may be complex). The optimizer determines predicate selectivities by a sampling strategy which is the only feasible way to support predicates with arbitrary data type operations. The optimizer is written in Prolog.

The GUI allows one to send commands and queries to a kernel and visualize the results. It supports both the executable level language and the SQL level. In the latter case it interacts with the optimizer to get a plan (executable query) which it then sends to the kernel. The GUI is extensible by so-called viewers which can offer their own methods to display data types. One of the available viewers (the so-called Hoese-Viewer) allows for a sophisticated representation of spatial data and for animation of spatio-temporal data types. This viewer is itself extensible to support further data types. The GUI is written in Java.

SECONDO is suitable for publishing research implementations for the following reasons:

- It offers clean concepts and interfaces for adding data structures and algorithms as type constructors and operators.

- A complete DBMS interface for data manipulation and querying is available at the executable level. Queries at this level are completely type checked. This is crucial for experiments as one can call query processing operators directly without any need to play tricks with a query optimizer. Very often it is also not clear how the new query operations could even be expressed in SQL. SECONDO allows one to focus on the query processing level, as many research papers do. We are not aware of any other DBMS that allows one to type in query plans directly.

- SECONDO provides simple and efficient concepts to deal with data types of widely varying size, from very small to very large. Basically small value representations are embedded into tuples, large representations are stored in separate records. This is transparent for the implementor of a data type. Such capabilities are crucial to deal with spatial or moving object types for which sizes are unpredictable.
- Query optimizer and GUI also support extensions as described above.
- Commands and queries of the executable level can be written into SECONDO scripts (text files with commands). Such scripts can be used to set up experimental data and to run experiments.

2.2 Support for Moving Objects

SECONDO is suitable for publishing implementations related to moving objects for the following reasons.

- Several algebras implement large parts of the data model of [12, 8], the fundamental data model for unconstrained movement (i.e. free space movement described by (x, y) coordinates). It provides representations for complete trajectories, but also for elements of trajectories (units, short linear pieces), as data types $\underline{mpoint}$ and $\underline{upoint}$. One can freely convert between a relation containing $\underline{mpoint}$ attributes (called the compact representation) and one having $\underline{upoint}$ attributes (unit representation).
- Beyond the trajectory types, the type system of [12] has a rich set of related standard, spatial, and time dependent types such as $\underline{mint}$, $\underline{mreal}$, and $\underline{mbool}$, representing time dependent integer, real, and boolean functions, respectively. SECONDO has implementations for all these types and for a large set of the related operations. They are crucial for formulating queries on moving objects.
- Furthermore, also a large part of the model for network-constrained movement [13] has been implemented. There are algebras offering the network data type (with a graph-based representation) and types for network based static and moving objects. At least all operations to execute the BerlinMOD benchmark on a network-based representation are available.
- R-trees and TB-trees are available for spatio-temporal indexing.
- The GUI provides visualizations and animation for all implemented data types of [12].
- A simple data set for moving objects is included in the SECONDO distribution, the *Trains* relation in database *berlintest*. By simple commands, this data set can be scaled up to arbitrary size. A further big and realistic data set *Cars* can be created by running the BerlinMOD benchmark (Section 3).

Hence there is a rich infrastructure into which new indexing or query processing methods can be integrated.

2.3 An Example: Nearest Neighbor Extension and Plugin

As an example, we consider a recent research paper whose implementation has been published as a SECONDO plugin [11]. The paper addresses the problem of finding the continuous k nearest neighbors to a query trajectory in a large set of stored trajectories. The result is a set of parts of trajectories.

Stored trajectories are given in unit representation and indexed in a 3D R-tree. For each node p of the R-tree, in preprocessing its coverage function is computed, which is the time dependent number of trajectories represented in p's subtree.

The solution uses a filter and refine strategy. In the filter step, the R-tree is traversed. Based on the coverage numbers of nodes, some nodes can be pruned. The filter step returns a stream of units ordered by start time which are guaranteed to contain the k nearest neighbors. The refinement step then determines the precise pieces of units forming the k nearest neighbors based on a plane sweep determining the k lowest distance curves.

The nearest neighbor extension benefits from the environment as follows:

- It uses the existing data types for moving point (trajectory) and for units.
- It uses R-trees and especially the bulkload facility to build its own index structure. It uses TB-trees to implement competing solutions.
- Coverage functions for R-tree nodes can be directly represented in a data type <u>*mint*</u> (moving integer). Hence an operation can be written in SECONDO that traverses an R-tree and returns a stream of tuples with coverage numbers, to be stored in a relation.
- Visualization of coverage functions and of R-tree nodes, available in SECONDO, has been crucial to determine an efficient shape of the R-tree index, which is constructed in the bulkload in a special way.
- Test data sets as scaled versions of *Trains* can be set up easily using SECONDO commands. A data set for long trajectories is created within the BerlinMOD benchmark.
- The algorithms for the filter step and for the refinement step can be defined as SECONDO operators *knearestfilter* and *knearest*, respectively. They are offered within a *NearestNeighborAlgebra*. A query using these operations can be written as follows (see [11] for a detailed explanation):

```
query UnitTrains25_UTrip UnitTrains25 UnitTrains25Cover_RecId
  UnitTrains25Cover knearestfilter[UTrip, train742, 5]
  knearest[UTrip, train742, 5] count;
```

- Competing algorithms [9, 10] have been implemented in the same environment as operators *greeceknearest* and *chinaknearest*, respectively. They can be run by queries:

```
query UTOrdered_RTreeBulk25 UTOrdered25 greeceknearest[UTrip, train742, 5] count;
query UnitTrains_UTrip_tbtree25 UnitTrains25 chinaknearest[UTrip,train742,5] count;
```

The plugin for [11] can be found at [3]. Scripts for repeating the experiments are available at [2].

3 BerlinMOD

3.0.0.1 Purpose Benchmarks have become the standard method to compare different DBMS. Each benchmark consists of a well defined data set and a workload, usually a set of queries. Though data structures, index structures and different operator implementations can be compared separately from other components, their impact on database performance becomes clearer when they are tested all together within in a real system. Benchmarks benefit researchers by simplifying the setup and description of experiments used to assess the efficiency of proposed inventions: The data properties are well defined and studied, the risk of introducing bias is minimized, and it becomes easier to repeat experiments. With BerlinMOD [7], we have proposed a benchmark to support research in the context of historic moving object database systems/ trajectory database systems.

3.0.0.2 Data Generation Though using moving object data (MOD) collected from the real world is widely considered preferrable, there exist severe problems with the amount and character of such data. Often, the trajectories are short, there are only few of them, or legal issues — such as data privacy or copyright — render their free use and distribution practically impossible. Therefore, BerlinMOD uses artificial MOD created by a data generator. The generator is implemented as a SECONDO script and allows for analysis and modification. With standard settings, data are obtained by long time observation (28 days) of 2,000 simulated vehicles' positions. The movements created are representative for employees' behaviour, who commute between their home and work location and do some additional trips (for visits, shopping, sports, etc.). The simulation uses geographic line data and tables with speed limits to create a traffic infrastructure network, and a combination of statistics on residential and employers' locations and regions describing statistical districts to create representative destinations for trips. In the standard version, real world data on Berlin is used to this end.

The data generator itself is not a standalone application, but a SECONDO script. It uses SECONDO's available complex data types (like relations, vectors, R-trees, graphs) and operators from different algebras to process the data from only three simple input tables and some parameters into arbitrary amounts of representative moving object data. Mainly to increase the performance, some parts of the data generation have been implemented as a special algebra module (SimulationAlgebra). The position information is enriched with some standard data, like a unique licence plate number, a type and a brand for each vehicle, thus allowing for more rich semantics in queries. Also, some time instant, time period, point, region and standard data is created for the purpose of workload generation with varying range parameters. Data can be directly used within SECONDO or exported to shapefile or CSV format for import to any third party DBMS.

Whereas the observed area is determined by the dimensions of the imported map data, both the observation period and the number of observed vehicles is scalable. With standard settings, vehicle positions are mapped to positions on the road network, but it is also possible to create noisy data. Each trip is created with individual deceleration and stop events, considering the geometry and character of roads (inferred from their speed limit) used and crossed.

The amount of data generated with standard settings is considerable: 19.45 GB for 292,693 trips — in average, each trajectory consists of 26,963 positions. However, users may scale the data or even produce MOD from an alternative scenario by simply replacing the original map data or changing parameters within the data generator script.

3.0.0.3 Queries BerlinMOD defines two sets of queries. BerlinMOD/R provides 17 range and point queries formulated in common English and SQL. The queries apply combinations of simple to rather complex standard, temporal, spatial and spatio-temporal predicates, operations and aggregations to the dataset. Both, selection and join queries are covered. The selection allows for testing a wide range of index structures, access methods and implementations of spatio-temporal operators.

The second query set, BerlinMOD/NN, aims at nearest neighbour queries. Nine queries combine k-NN, reverse NN, and aggregated NN queries with both, static and moving query objects and candidate objects. While solutions to some of these combinations have already been proposed, others are still uncovered in the context of trajectory database systems. This part of BerlinMOD also gives a good prospect of how BerlinMOD can be extended to address new aspects of data processing while using standardized, existent data.

You can obtain BerlinMOD from a dedicated web page [1] on the SECONDO web site. As SECONDO itself, BerlinMOD is free and open software you are invited to use, examine, extend, publish and propagate free of all charges.

4 Plugins

To publish a plugin for a SECONDO extension, the implementor fills in an XML-file. Basically one needs to specify which SECONDO version is used, which dependencies to other algebra modules exist, and which extensions are provided in which files. Different XML tags describe the different kinds of extensions possible such as algebra modules, viewers, display classes for the Hoese viewer, and optimizer extensions. Afterwards, all files needed for the extension and the describing XML file are packed into a zip file which can be published at the author's web site. More details are given at the plugin web pages [3].

The reader of a paper referring to a plugin needs to have a working SECONDO installation which can be obtained from the SECONDO web site [4]. She gets the plugin *X.zip* from the author's web site. The SECONDO system contains an installer for plugins. The command `secinstall X.zip` integrates the components of the plugin into the source code of the SECONDO system. Afterwards a call of `make` builds the system with the extension.

On the plugin website [3], we provide several plugins for download. These include implementations of a TB-Tree structure [14], an algebra containing X-Trees and M-Trees [5, 6], an algebra together with display classes for periodic moving objects, an algebra providing operators for finding the k nearest neighbours to an object within a set of static or moving objects, and an algebra together with optimizer extensions for querying moving objects by their movement profile. The latter two are based on [11] and [15], respectively. The papers demonstrate how a publication can profit from the work invested into making the implementation available as a plugin.

References

[1] BerlinMOD. http://dna.fernuni-hagen.de/secondo/BerlinMOD.html.

[2] Scripts to execute the experiments of the knn paper. http://dna.fernuni-hagen.de/papers/KNN/knn-experiment-script.zip.

[3] Secondo Plugins. http://dna.fernuni-hagen.de/Secondo.html/start_content_plugins.html.

[4] Secondo Web Site. http://dna.fernuni-hagen.de/Secondo.html/.

[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *VLDB*, pages 28–39, 1996.

[6] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[7] C. Düntgen, T. Behr, and R.H. Güting. BerlinMOD: a benchmark for moving object databases. *VLDB J.*, 18(6):1335–1368, 2009.

[8] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *SIGMOD*, pages 319–330, 2000.

[9] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *GeoInformatica*, 11(2):159–193, 2007.

[10] Y. Gao, C. Li, G. Chen, Q. Li, and C. Chen. Efficient algorithms for historical continuous *k* nn query processing over moving object trajectories. In *APWeb/WAIM*, pages 188–199, 2007.

[11] R. H. Güting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. Technical report, University of Hagen, Informatik-Report 352, 2009, *VLDB J.*, to appear.

[12] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and quering moving objects. *ACM TODS*, 25(1):1–42, 2000.

[13] R. H. Güting, V. Teixeira de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *VLDB J.*, 15(2):165–190, 2006.

[14] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *VLDB*, pages 395–406, 2000.

[15] M.A. Sakr and R.H. Güting. Spatiotemporal pattern queries. Technical report, University of Hagen, Informatik-Report 352, 2009.

Real-Time Traffic Information Management using Stream Computing

Alain Biem, Eric Bouillet, Hanhua Feng, Anand Ranganathan, Anton Riabov, Olivier Verscheure
IBM TJ Watson Research Center, NY, USA
{biem, ericbou, hanhfeng, arangana, riabov, ov1}@us.ibm.com

Haris Koutsopoulos, Mahmood Rahmani
The Royal Institute of Technology, KTH, Stockholm, Sweden
{hnk,mahmoodr}@kth.se

Barış Güç
ETH, Zurich, Switzerland
baris@student.ethz.ch

Abstract

With the widespread adoption of location tracking technologies like GPS, the domain of transportation information management has seen growing interest in the last few years. In this paper, we describe a stream processing infrastructure for processing large volumes of sensor data in real time to derive useful traffic and travel planning information. We have used this infrastructure to process floating car data for the city of Stockholm in real-time. Our findings show that there is a great need for real-time traffic information management because of the tremendous variability in traffic conditions in a city like Stockholm. Also, our stream processing infrastructure can help meet this need by supporting the development of applications that can process large volumes of GPS and other data on a distributed cluster of machines.

1 Introduction

Intelligent Transportation Systems (ITS) have brought many advances in the transportation management field. An important development is the emergence and installation of sensor technologies for the collection of various types of data on the state of the transport system. GPS is an excellent example of this new generation of sensors that has the potential to provide high quality traffic data for real time traffic monitoring and management, as well as planning, policy, and applications, at a relatively low cost. An important characteristic of this new source is that it includes a fair amount of floating car data (FCD). FCD represent the location of vehicles collected by mobile sources, such as GPS devices installed in vehicles or cellular phones. This raw data can be sent to a central facility, where it can be processed in real-time.

In this paper, we briefly describe some of our recent work in supporting real-time Traffic Information Management using a stream computing approach. This work was made possible by access to GPS data from some taxis and trucks in the city of Stockholm. We highlight some of our findings on traffic variability in the city of Stockholm. We also show how we have used IBM's System S stream processing platform for the purpose of real-time traffic information management. We have developed applications on this platform that process real-time GPS data, generate different kinds of real-time traffic statistics, and perform customized analyses in

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

response to user queries. Examples of customized analyses include continuously updated speed and traffic flow measurements for all the different streets in a city, traffic volume measurements by region, estimates of travel times between different points of the city, stochastic shortest-path routes based on current traffic conditions, etc. Our system can handle large volumes of incoming GPS data. For instance, on a cluster of four x86 blade servers, it can process over 120000 incoming GPS points per second, combine it with a map containing over 600,000 links, continuously generate different kinds of traffic statistics and answer user queries.

2 Need for Real-Time Traffic Information

Obtaining real-time traffic information is extremely valuable. In fact, our experiments using the GPS data set from the city of Stockholm reveals that traffic in the city varies significantly during different time scales. Figures 1 and 2 provide a glimpse into the highly variable traffic situation. These figures illustrate the immense benefits that can be obtained for commuters, city agencies, commercial fleet operators and other parties if they have access to real time traffic information and predictions. We shall describe the actual data and the processing of the data to derive these statistics in Section 3.

Figure 1 shows the variability of the travel times between the center of the city of Stockholm and Arlanda Airport. These travel times were obtained by observing the same vehicle at the two locations within a certain time interval. The travel times are grouped by day of the week and time period (blue for the morning and red for the afternoon period). The results indicate that average (shown by dot) and standard deviation (std, shown by bold vertical lines) of travel times varies day-by-day. For example, as expected, the average is less during weekends compared to weekdays, but there is also variability among weekdays. Additionally, the travel time depends on the time of the day, both in terms of average and variability. For example, the average and standard deviation travel time in the afternoon (red) is increased by few minutes compared to the evening (blue).

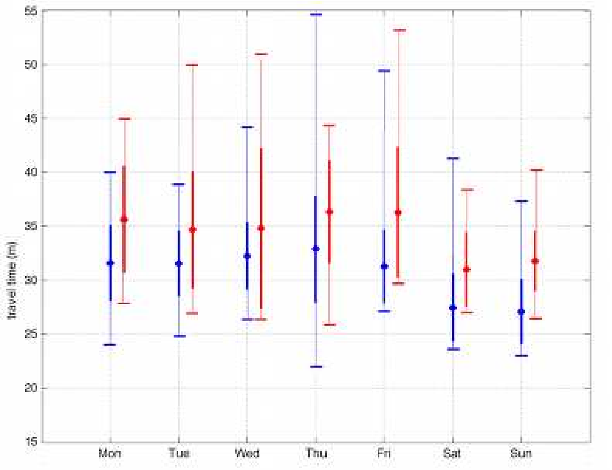


Figure 1: Travel time variability for the Stockholm-Arlanda route. Average (dot), standard deviation (bold line), and min/max (bar) of travel time. 7:00-9:00 (blue) and 16:00-18:00 (red).

Another experiment we ran with our GPS data set was to first derive the average speed on different road links at different 5 minute intervals of time during the day. We then used this average speed information to compute time-dependent shortest paths between various pairs of points in the city. We calculated the shortest paths between two points for 50 consecutive departure times between 6 AM and midnight, spaced 20 minutes apart. In this way, we can get an idea of how frequently the shortest path between two points on the road changes during the day. Each shortest path calculation uses time-varying link travel time information, where the estimated travel time on each link for each 5 minute interval during the day was calculated based on historical averages.

In Figure 2, we visualize an origin-destination pair where the time and traffic dependent shortest path changes frequently during the course of the day. The 50 shortest paths found for different departure times are displayed between the two points. The large number of changes for this pair is due to the fact that the points are in the central area of the city where there is a large number of links with traffic updates and due to the existence of many distinct alternative roads between the origin and the destination. The map shows how the shortest path switches between different highways and local roads during the day, and how smaller streets can be used as shortcuts instead of highways in some periods.

This figure shows that the best route between two points in the city can change very often, and in much more diverse ways than may have been imagined. Hence, making use of real time traffic information and good traffic prediction algorithms can result in massive savings of time and energy on the part of drivers.

3 Stream Processing Infrastructure

A key feature in our work has been the use of stream processing for the purpose of real time traffic information management. In particular, we have used System S [4] [1], which is an IBM research platform that supports high performance stream processing. It has been used in a variety of sense-and-respond application domains, from environmental monitoring to algorithmic trading. It offers both language and runtime support for improving the performance of streaming applications via a combination of optimized code generation, pipelining and parallelization. It supports a component-based programming model that simplifies the development of complex applications.

Figure 2: Origin-destination pair with a large number of different shortest paths. 50 shortest paths between 6 AM and midnight for this origin-destination pair are displayed. The paths are drawn with offsets for better visualization. (©2010 Google - Map data ©2010 Tele Atlas)

System S supports structured as well as unstructured data stream processing and can be scaled to a large number of compute nodes. The runtime can execute a large number of long-running jobs (queries) that take the form of data-flow graphs. A data-flow graph consists of a set of operators connected by streams, where each stream carries a series of Stream Data Objects (SDOs). The operators communicate with each other via their input and output ports, connected by streams. The operator ports as well as the streams connecting them are typed.

System S supports a declarative language called SPADE [3] to program stream-processing applications and to define the data-flow graph. SPADE supports a modular, component-based programming model, which allows reuse, extensibility and rapid prototyping. It supports a toolkit of all basic stream-relational operators with rich windowing semantics. It allows extending the set of built-in operators with user-defined ones, programmable in either C++ or Java. It also supports a broad range of stream adapters used to ingest data from outside sources and publish data to outside destinations, such as network sockets, relational and XML databases, etc. It also allows developing applications that offer high-availability through replicated processing and operator checkpointing.

System S includes a scheduler component that decides how best to partition a data-flow graph across a distributed set of physical nodes [8]. The scheduler uses the computational profiles of the operators, the loads on the nodes and the priority of the application in making its scheduling decisions.

3.1 The Raw Data

We obtained historical GPS data traces from Trafik Stockholm [7] for the year of 2008. This data included traces from about 1500 taxis and 400 trucks that plied the streets of Stockholm. In total, there was about 170 million GPS probe points for the whole year. Each taxi produces a GPS probe reading once every 60 seconds that includes taxi identification and location information. Also, for privacy reasons, taxis produce fewer readings

[1] System S is the basis for the IBM InfoSphere Streams product

when they are carrying passengers. Trucks use more recent and more accurate GPS devices, that produce readings once every 30 seconds and include identification location, speed and heading information. The data rate for the whole city was over 1000 GPS readings per minute. However, our system can handle much larger input rates.

In addition, we are now receiving real-time data on GPS from taxis as well. The data rate is still in the order of 1000 GPS readings per minute. In fact, Figure 1 was derived from this real-time data feed over a 4 month period from Dec 2009 to Mar 2010, and Figure 2 was derived from the historical GPS data from all of 2008.

The Stockholm city road network which has 80735 polylines (i.e. road segments) and 37458 nodes (i.e. intersections). The maximum link length on the network is 10756 meters while the average is 142 meters. These 80735 polylines translate into over 600,000 individual line segments.

3.2 Overall Application Description

Having large numbers of vehicles sending real-time GPS data for the city allows us to create a picture of the traffic condition in time and space [6]. We now describe the stream processing applications that allow us to come up with the traffic information, as well as provide various value-added services on top of the basic information.

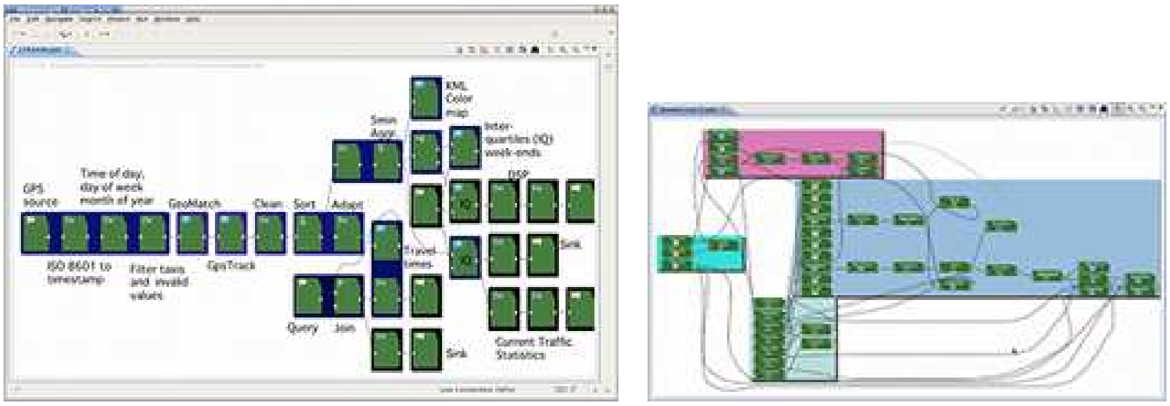
The application processes the data in three distinct phases. The first phase consists of real-time processing of the data. This includes obtaining, cleaning, de-noising, and matching the GPS data to the underlying road network or specified regions. In the second phase, the data is aggregated to produce traffic statistics per link and per time interval. The traffic statistics are in the form of medians and quartiles of vehicle speeds and vehicle counts on the link or region for the time interval. In the final phase, we make use of the statistics to compute different kinds of derived information such as the estimated travel times and shortest paths between different parts of the city. The final outputs can be sent to the user on different kinds of visualization platforms such as Google Earth, or may be stored in a database for additional offline analysis. Further details of this process is described in [1]. A video describing the prototype is available at [5].

For the shortest path calculation, we use the method described in [2] to adapt A* algorithm for time-dependent networks. The A* algorithm is an improvement over Dijkstra algorithm for the one origin, one destination shortest path problem and it utilizes a heuristic function that decreases the search space. In our implementation, A* is adapted using the following modification: as paths are extended with links during the execution of A*, time is advanced and therefore future delay values of links are used as link costs. In addition, the algorithm is adapted to accept continuous changes in the estimated travel times for each link based on the current and predicted traffic conditions.

Figure 3(a) shows a screenshot of the deployed application. In the figure, boxes represent operators, interconnections represent data streams, and the resulting topology represents the entire application flow-graph. It also shows how some operators are fused together to form a PE (Processing Element), represented as large dark background rectangles that contain one or more individual operators. The purpose of fusing operators is to reduce the data transfer latency by having these operators be part of the same process with a shared address space. This fusion of operators is one of the many performance enhancing optimizations supported by the System S platform.

Figure 3(b) shows how those PEs are distributed across various hosts (nodes). In this example, the distribution is based on instructions in the SPADE program assigning different operators to various nodes, but it can also be done automatically by the System S scheduler.

This particular application is designed to deal with GPS data, but can accommodate other sources to better estimate and predict traffic conditions. These include induction loops, weather data, road incident information, video cameras, etc. The addition and processing of various types of data is a particular challenge for real time transportation information management. The System S platform, with its component based and modular programming model, does simplify this process and allows incrementally adding new kinds of data and processing.



(a) Application flow graph of operators, with labels describing the operations performed by different groups of operators.

(b) Application flow graph showing PE within hosts.

Figure 3: Different visualizations of the application.

4 Conclusion

In this paper, we have motivated the need for real time traffic information management using examples of traffic variability in the city of Stockholm. We have also briefly described a scalable stream processing approach for supporting real time traffic information management. Our implementation is based on IBM's System S platform, which is well suited to deal with scalability and adaptability challenges associated with real-time traffic information management. As part of future work, we are investigating several enhancements to our current implementation, including traffic prediction, multi-modal travel planning and multi-sensor data fusion.

Acknowledgements: The authors would like to thank Inga-Maj Eriksson from the Swedish Transport Administration and Tomas Julner from Transport Administration and Trafik Stockholm for their support, feedback, and provision of the data for this study, and Erling Weibust of IBM for facilitating the IBM/KTH collaboration.

References

[1] A. Biem, E. Bouillet, H. Feng, H. Koutsopoulos, C. Moran, A. Ranganathan, A. Riabov, and O. Verscheure. IBM InfoSphere Streams for Scalable, Real-Time, Intelligent Transportation Services. In SIGMOD 2010

[2] I. Chabini and S. Lan. Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Trans on Intelligent Transportation Systems*, 3(1):60–74, 2002.

[3] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo. SPADE: the System S declarative stream processing engine. In *SIGMOD 2008*, pages 1123–1134, 2008.

[4] IBM InfoSphere Streams. http://www-01.ibm.com/software/data/infosphere/streams/.

[5] IBM InfoSphere Streams. InfoSphere Streams enables smarter transportation at the city of Stockholm. http://www.youtube.com/watch?v=i2yCfLQV6M8.

[6] R. Kuehne, R.-P. Schaefer, J. Mikat, K. Thiessenhusen, U. Boettger, and S. Lorkowski. New approaches for traffic management in metropolitan areas. In *Proceedings of IFAC CTS Symposium*, 2003.

[7] Trafik Stockholm. http://www.trafikstockholm.com.

[8] J. Wolf et al. SODA: an optimizing scheduler for large-scale stream-based distributed computer systems. In *Middleware*, pages 306–325, 2008.

Spatio-Temporal Stream Processing in Microsoft StreamInsight

Mohamed Ali[1] Badrish Chandramouli[2] Balan Sethu Raman[1] Ed Katibah[1]

[1]Microsoft Corporation, One Microsoft Way, Redmond WA 98052
[2]Microsoft Research, One Microsoft Way, Redmond WA 98052
{mali, badrishc, sethur, edwink}@microsoft.com

Abstract

Microsoft StreamInsight is a platform for developing and deploying streaming applications. StreamInsight embraces a temporal stream model to unify and further enrich query language features, handle imperfections in event delivery and define consistency guarantees on the output. With its extensibility framework, StreamInsight enables developers to integrate their domain expertise within the query pipeline as user defined functions, operators and aggregates. Also, the Microsoft SQL Server Spatial Library delivers comprehensive spatial support that enables organizations to seamlessly consume, use, and extend location-based data.

This paper covers two approaches to support spatio-temporal stream processing in StreamInsight. First, the paper describes the ongoing effort at Microsoft SQL Server to bring together the temporal aspect of StreamInsight and the spatial support of the SQL Spatial Library, through the extensibility framework, to deliver an end-to-end solution for location-aware and geospatial data streaming applications. Second, the paper provides the future vision for supporting spatial attributes natively within the pipeline of the stream query processor.

1 Introduction

Microsoft StreamInsight (StreamInsight, for brevity) is a platform for stream query processing. Thanks to its real-time low-latency output, StreamInsight monitors, analyzes and correlates stream data from multiple sources to extract meaningful patterns and trends. StreamInsight adopts a deterministic stream model that leverages a temporal algebra as the underlying basis for processing long-running continuous queries. In most streaming applications, continuous query processing demands the ability to cope with high input rates that are characterized by imperfections in event delivery (i.e., incomplete or out-of-order data). StreamInsight is architected to handle imperfections in event delivery and to provide consistency guarantees on the resultant output. Such consistency gurantees place correctness measures on the output that has been generated so far, given the fact that late and out-of-order stream events are still in transient.

Data stream systems have been widely adopted across multiple business domains. Because domain expertise is the outcome of focused experience in a specific business sector over years, it is hard to expect that a

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

single streaming engine (out-of-the-box without any specialization) can address the requirements of all domains. However, a streaming system is expected to have an extensibility mechanism that seamlessly integrates domain-specific logic into the query pipeline. Hence, StreamInsight has been designed to be an extensible system that accepts and executes user defined modules (*UDMs*) as part of the continuous query plan.

Meanwhile, Microsoft SQL Server Spatial Library [1] (SQL Spatial Library, for brevity) provides an easy to use, robust and high performance environment for persisting and analyzing spatial data. SQL Spatial Library provides data type support for point, line and polygon objects. Also, various methods are provided to handle these spatial data types. SQL Spatial Library adheres to the *Open Geospatial Consortium Simple Feature Access* specification [2] and is provided as part of the SQL Server Types Library.

This paper tackles two approaches for the spatio-temporal stream processing within StreamInsight: an extensibility approach and a native support approach. The extensibility approach combines the values of the StreamInsight extensibility framework and the SQL Spatial Library by giving the UDM writers the ability to invoke the library methods within their code. Alternatively, the native support approach deals with spatial attributes as first class citizens, reasons about the spatial properties of incoming events and, more interestingly, provides consistency guarantees over space as well as time. This paper overviews both approaches and highlights the pros and cons of each approach. The remainder of this paper is organized as follows. Section 2 provides basic background on StreamInsight. Section 3 presents the extensibility approach while Section 4 presents the native support approach for stream processing . Section 5 concludes the paper.

2 Background

A *Data Stream Management System* (*DSMS*) [3, 7, 8, 13] enables applications to issue long-running *continuous queries* (*CQs*) that monitor and process streams of data in real time. DSMSs are used for efficient real-time data processing in many applications. In this paper, we focus on Microsoft StreamInsight [4, 6], which is a DSMS based on the CEDR [5] research project.

2.1 Streams and Events

A *physical stream* is a sequence of events. An event $e_i = \langle p, c \rangle$ is a notification from the outside world that contains: (1) a payload $p = \langle p_1, \ldots, p_k \rangle$, and (2) a control parameter c that provides metadata. The control parameter includes an event generation time, and a duration that indicates the period of time over which an event can influence output. We capture these by defining $c = \langle \mathsf{LE}, \mathsf{RE} \rangle$, where the interval $[\mathsf{LE}, \mathsf{RE})$ specifies the period (or lifetime) over which the event contributes to output. The left endpoint (LE) of this interval, also called start time, is the application time of event generation, also called the event timestamp. Assuming the event lasts for x time units, the right endpoint of an event, also called end time, is simply $\mathsf{RE} = \mathsf{LE} + x$.

2.1.0.4 Compensations StreamInsight allows users to issue compensations (or corrections) for earlier reported events, by the notion of retractions [5, 10, 11], which indicates a modification of the lifetime of an earlier event. This is supported by an optional third control parameter $\mathsf{RE}_{\mathsf{new}}$, that indicates the new right endpoint of the corresponding event. Event deletion (called a full retraction) is expressed by setting $\mathsf{RE}_{\mathsf{new}} = \mathsf{LE}$ (i.e., zero lifetime).

2.1.0.5 Canonical History Table A *Canonical History Table* (*CHT*) is the logical representation of a stream. Each entry in a CHT consists of a lifetime (LE and RE) and the payload. All times are application times, as opposed to system times. Thus, StreamInsight models a data stream as a time-varying relation, motivated by early work on temporal databases by Jensen and Snodgrass [9]. Table 3 shows an example CHT. This CHT can be derived from the actual physical events (either new inserts or retractions) with control parameter

ID	LE	RE	Payload
e_0	1	5	P_1
e_1	4	9	P_2

Table 3: Canonical History Table.

ID	Type	LE	RE	$\mathsf{RE_{new}}$	Payload
e_0	Insertion	1	∞	-	P_1
e_0	Retraction	1	∞	10	P_1
e_0	Retraction	1	10	5	P_1
e_1	Insertion	4	9	-	P_2

Table 4: Physical stream corresponding to CHT.

$c = \langle \mathsf{LE}, \mathsf{RE}, \mathsf{RE_{new}} \rangle$. For example, Table 4 shows one possible physical stream with an associated logical CHT shown in Table 3. Note that a retraction event includes the new right endpoint of the modified event. The CHT (Table 3) is derived by matching each retraction in the physical stream (Table 4) with its corresponding insertion and adjusting RE of the event accordingly.

2.1.0.6 Detecting Time Progress We need to ensure that an event is not arbitrarily out-of-order; this is realized using time-based punctuations [5, 12, 15]. A time-based punctuation is a special event that is used to indicate time progress. These punctuations are called *Current Time Increments* (or *CTIs*) in StreamInsight. A CTI is associated with a timestamp t and indicates that there will be no future event in the stream that modifies any part of the time axis that is earlier than t. Note that we could still see retractions for events with LE less than t, as long as both RE and $\mathsf{RE_{new}}$ are greater than or equal to t.

2.2 Stream Queries and Operators

A CQ consists of a tree of operators, each of which performs some transformation on its input streams and produces an output stream. In StreamInsight, queries are expressed in LINQ [14]. StreamInsight operators are well-behaved and have clear semantics in terms of their effect on the CHT. This makes the underlying temporal operator algebra deterministic, even when data arrives out-of-order. Data enters the streaming system via *input adapters*, which convert external sources into events that can be processed by the streaming system. Output events exit the system via *output adapters*.

There are two main classes of operators: span-based and window-based. A span-based operator accepts events from an input, performs some computation for each event, and produces output for that event. Examples of span-based operators include filter, project, and temporal join. On the other hand, aggregation operators such as Count, Top-K, Sum, etc. report a result (or set of results) for every unique window, i.e., they are window-based. The result is computed using all events that belong to that window. StreamInsight supports several types of windows: snapshot (sliding), hopping, tumbling, and count-based windows.

Further, one stream can be output to multiple operators using an operator called multicast, while multiple streams are merged using a union operator. StreamInsight allows per-group computation using an operation called Group&Apply, where the same subplan (called the apply branch) is applied in parallel for every group (defined by a grouping key) in a stream. The results of all the groups are merged (using union) as the final operator output. In addition, StreamInsight supports user-defined operators to express custom computations; these are discussed next.

3 The Extensibility Approach

As shown in Figure 1, there are three distinct entities that collaborate to extend a streaming system. The *user defined module* (*UDM*) *writer* is the domain expert who writes and packages the code that implements domain-specific operations as libraries. The *query writer* invokes a UDM as part of the query logic. A query is expected to have one or more UDMs wired together with standard streaming operators (e.g., filter, project, joins). Note that multiple query writers may leverage the same existing repository of UDMs for accomplishing specific

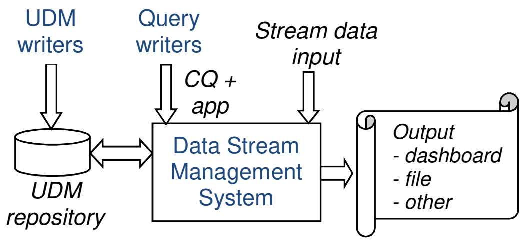


Figure 1: Entities in a streaming extensibility model.

business objectives. The *extensibility framework* is the component that connects the UDM writer and the query writer. The framework executes the UDM logic on demand based on the query to be executed. Thus, the framework provides convenience, flexibility, and efficiency for both the UDM writer and the query writer.

StreamInsight supports three fundamental types of UDMs to the system — *user-defined functions* (*UDFs*), *user-defined aggregates* (*UDAs*), and *user-defined operators* (*UDOs*). UDFs are method calls with arbitrary parameters and a single return value. They can be used wherever expressions (span-based stream operators) occur: filter predicates, projections, join predicates, etc. A UDA is used on top of a window specification (e.g., hopping, snapshot, or count-based window) to aggregate the events contained in each window. A UDA processes a set of events (in a window) and produces a scalar aggregation result (e.g., integer, float, string, etc). UDOs are also used on top of a window, but the result is a set of events with timestamps rather than a scalar value. Note that based on the type of extension, UDMs surface in the StreamInsight LINQ programming model either as method calls (in case of span-based operators) or as extension methods (in case of window-based stream operators).

3.1 Integrating the SQL Spatial Library within the Stream Query Processor

The SQL Spatial Library [1] provides methods to perform spatial operations on spatial data types. Thanks to the extensibility framework of StreamInsight, UDMs that perform intersection, containment, nearest neighbor and shortest route operations are implemented by invoking the appropriate methods from the SQL Spatial Library. Combining the StreamInsight extensibility model and the existing SQL Spatial library provides a solution for spatio-temporal stream processing, increases the value of existing (or *out-of-the-box*) modules, and reduces the cost to develop spatial-oriented streaming applications. Note that the SQL Spatial Library is *not* designed with the continuous stream processing paradigm in mind and, hence, is non-incremental by nature. An appealing future direction is to port the SQL Spatial Library to the streaming domain with the *incremental single-pass* model of stream processing in mind. For example, an intersection query is evaluated incrementally at time $T+1$ by reusing the computed state at time T as the moving object changes its location from time T to time $T+1$.

3.2 Breaking the Optimization Boundaries

Since a UDM is a black box to the query optimizer, the UDM stands as an optimization boundary in the query pipeline. However, there are two approaches to break the optimization boundary in the extensibility approach. The first approach is to work hand-in-hand with the UDM writer who has the option to provide several properties about the UDM through well-defined interfaces to the cost-based query optimizer. Examples of these properties include the selectivity and the expected CPU load (cycles) per input tuple. The optimizer reasons about these properties and shoots for optimization opportunities, e.g., via query plan reorganization and operator migration (in case of multiple StreamInsight instances running in parallel). In the second approach, the system automatically instruments the UDM to measure its average throughout and selectivity. While the second approach relieves the user from specifying the operator properties, the system goes through a learning period during which sub-optimal performance may be observed. With either approach, there is nothing special about spatio-temporal data streams from the extensibility framework's point of view. As we directly leverage the ex-

(a) (b) (c)

Figure 2: An example of native support for spatio-temporal streams.

tensibility framework in order to break the optimization boundary, spatio-temporal stream processing benefits from these optimizations *without* the need to specialize the system for spatio-temporal processing. This solution highlights the value and generality of the extensibility framework.

4 Native Support for Consistent Spatio-temporal Stream Processing

As described in Section 2, the canonical history table maintains the temporal endpoints LE and RE of the event's validity interval over time. Consider a spatial application where we wish to track the movement of objects (e.g., cars). The naive technique is to let the object generate an event periodically as its spatial location (detected using GPS, for example) changes. However, this can result in considerable network traffic and does not scale well as the number of objects increases.

We can instead augment the control parameters with the following information: (1) two location markers SL and DL to denote the start and destination locations of the moving object between times LE and RE, (2) the *route selection policy*, that is the route the system assumes the object takes to travel from the starting location to the destination. Each route selection policy optimizes for one or more criteria (e.g., shortest distance, shortest travel time, highway avoidance, etc) and deterministically decides on the route between the endpoints. (3) detailed temporal information of the planned trip to compute the time at which the object hits a specific point on the route — this could use a simple model such as assuming constant speed over the entire route.

Given the spatial and temporal attributes that describe the object's route over time (for every moving object in the system), the system has the ability to speculate and predict the state of the monitored environment at any point in time. For example, in a traffic management scenario, the system answers queries about the past, current, and future road conditions. Further, it suggests the best driving directions for newly added vehicles by taking future road conditions into consideration. Note that as long as the vehicle is on track, i.e., following the route planned by the system according to the expected speed, there is no need for the vehicle to transmit regular events to the system, which results in reducing transmission load over the wireless network. However, if the vehicle changes its route selection policy, makes an unexpected turn, or stops for some time, the vehicle generates retraction and insertion events to adjust its path. In response to the retraction event, the system updates the result of its CQs and possibly generates compensation events or new speculative output. Further, we could define a *spatio-temporal algebra* with new streaming operators that natively take location into consideration; for example, we may add a spatio-temporal *left-semi-join* operator that accepts a proximity metric and outputs events related to the left input object only when it overlaps in time as well as space (within the proximity metric) with a matching object on the right input. Note that such native support exposes optimization opportunities beyond those possible with black-box approaches.

As a concrete example, consider a CQ that reports the number of cars moving over "*Microsoft Way*". In Figure 2(a), the DSMS receives an insertion event that denotes the intent of car 1 to travel from point *A* to point *B* at time T. In response, it evaluates the query output to be one car traveling over *Microsoft Way* for a specific time interval, i.e., the expected time duration when car 1 is present on Microsoft Way. In Figure 2(b), the DSMS receives another insertion that denotes the intent of car 2 to travel from point *C* to point *B* at time T'. The system modifies the lifetime of the earlier event accordingly, and generates a new event for the duration when 2 cars are present on the road segment. In Figure 2(c), the DSMS receives a retraction that denotes a change in the intent of car 2 from destination *B* to destination *D*. Consequently, it retracts the previously generated event and reverts the count back to one. Although this example shows a simple query over a spatio-temporal stream of two objects, the concept is generalizable to larger road networks and more objects.

5 Conclusions

Microsoft StreamInsight is a high-performance platform for developing streaming applications. In this paper, we presented two approaches to support spatio-temporal data streams in StreamInsight. The first approach utilizes the extensibility framework of StreamInsight to invoke methods from the Microsoft SQL Server Spatial Library. The second approach supports the spatial attributes of moving objects natively as system attributes. The first approach increases the value of existing libraries and components, and gives spatio-temporal stream processing the ability to piggyback on optimizations applied within the extensibility framework as it evolves over time. The second approach extends the temporal algebra adopted by StreamInsight in the spatial direction to provide consistency guarantees over space as well as time, in addition to greater optimization opportunities.

6 Acknowledgments

We would like to thank Michael Kallay for his feedback and his help in using the SQL Spatial Library.

References

[1] SQL Server Spatial Libraries, http://www.microsoft.com/sqlserver/2008/en/us/spatial-data.aspx.

[2] Open Geospatial Consortium, http://www.opengeospatial.org/standards/sfa.

[3] D. Abadi et al. The design of the Borealis stream processing engine. In *CIDR*, 2005.

[4] M. Ali et al. Microsoft CEP Server and Online Behavioral Targeting. In *VLDB*, 2009.

[5] R. Barga et al. Consistent streaming through time: A vision for event stream processing. In *CIDR*, 2007.

[6] B. Chandramouli, J. Goldstein, and D. Maier. On-the-fly progress detection in iterative stream queries. In *VLDB*, 2009.

[7] S. Chandrasekaran et al. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[8] C. Cranor et al. Gigascope: A stream database for network applications. In *SIGMOD*, 2003.

[9] C. Jensen and R. Snodgrass. Temporal specialization. In *ICDE*, 1992.

[10] R. Motwani et al. Query processing, approximation, and resource management in a DSMS. In *CIDR*, 2003.

[11] E. Ryvkina et al. Revision processing in a stream processing engine: A high-level design. In *ICDE*, 2006.

[12] U. Srivastava and J. Widom. Flexible time management in data stream systems. In *PODS*, 2004.

[13] StreamBase Inc. http://www.streambase.com/.

[14] The LINQ Project. http://tinyurl.com/42egdn.

[15] P. Tucker et al. Exploiting punctuation semantics in continuous data streams. *IEEE TKDE*, 2003.

Call for Participation
VLDB 2010
36th International Conference on
Very Large Data Bases
Singapore : 13 to 17 Sept 2010
Grand Copthorne Waterfront Hotel
http://www.vldb2010.org

VLDB is a premier annual international forum for data management and database researchers, vendors, practitioners, application developers, and users. It will cover current issues in data management, database and information systems research. Data management and databases remain among the main technological cornerstones of emerging applications of the twenty-first century.

The main tracks of the conference, featuring research talks, tutorials and demonstrations, are held from 14th to 16th September. This year's keynote speakers are Divesh Srivastava, from AT&T and Paul Matsudaira, from NUS. VLDB 2010 hosts six tutorials: "Big Data and Cloud Computing: New Wine or just New Bottles?" by Divyakant Agrawal, Sudipto Das, and Amr El Abbadi, "Data Management and Mining in Internet Ad Systems" by S. Muthukrishnan, "Similarity Search and Mining in Uncertain Databases" by Hans-Peter Kriegel, Matthias Renz, Thomas Bernecker, and Andreas Zuefle, "Event Processing - Past, Present, Future" by Opher Etzion, " Similarity Searching: Indexing, Nearest Neighbor Finding, Dimensionality Reduction, and Embedding Methods in Multimedia Databases" by Hanan Samet, and "Distributed Caching Platforms" by Anil Nori.

Before and after the main conference, VLDB is hosting a PhD workshop and 10 thematic workshops. The PhD workshop, the 8th International Workshop on Quality in Databases (QDB), the Workshop on Personalized Access, the Workshop on Profile Management, and Context Awareness in Databases (PersDB), the 7th International Workshop on Data Management for Sensor Networks (DMSN), the 4th International Workshop on Management of Uncertain Data (MUD), the 1st International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (ADM) and the Workshop on Enabling Real-Time Business Intelligence (BIRTE) take place on 13th September. The Workshop on Semantic Data Management (SemData), the 7th VLDB Workshop on Secure Data Management (SDM), the 7th International XML Database Symposium (XSym) and the TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC) take place on 17th September.

Singapore is situated at the southern tip of the Malay Peninsula in South-East Asia. It provides all the modern facilities conducive to effective work: a safe environment, comfortable accommodation, efficient transportation and effective information and communication infrastructure. It is a vibrant Science & Technology hub where several universities and numerous public and corporate research institutes perform high-quality research over a broad range of disciplinary and cross-disciplinary areas. Singapore is a small and prosperous cosmopolitan nation that offers to the visitor the cultural diversity of its four main ethnic groups, namely, Chinese, Malays, Indians and Eurasians. With museums, theme parks, beaches, nature parks and two new integrated resorts as well as bustling shopping and dining heavens, Singapore is an excellent place to unwind. It is also just a short and cheap hop away from Malaysia, Indonesia, Thailand, Cambodia and other exciting destinations. The combination of an excellent technical programme and a meticulously-planned event in an unique cultural setting will make Singapore an unforgettable VLDB experience.

The conference co-organizers---National University of Singapore, Nanyang Technological University, and Singapore Management University---are looking forward to your visit.

VLDB 2010 is sponsored and supported by Microsoft, Oracle (Platinum sponsors), Google, IBM, SAP, Sybase (Gold sponsors), ADSC, Yahoo, TPC, HP (Silver sponsors), Sun (supporter) A*Star, Singapore Tourism Board (supporting organizations), KIISE, COMAD, DBSJ and CCF (supporting societies).

On-line registration is available at the conference website from June 12, 2010.

If you choose to stay at the conference hotel or at one of the supporting hotels, download the form from the conference website for bookings at negotiated conference rates and fax or mail it to the address indicated in the form. For further information about Arriving in Singapore and Sightseeing in Singapore, visit the conference website and Uniquely Singapore (www.visitsingapore.com), the comprehensive official tourist information site of Singapore Tourism Board (www.stb.com.sg). You can find detailed information about visa requirements at http://www.ica.gov.sg.

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398