Managing Moving Objects on Dynamic Transportation Networks

Zhiming Ding Ralf Hartmut Güting
Praktische Informatik IV, Fernuniversität Hagen, D-58084 Hagen, Germany
{zhiming.ding, rhg}@fernuni-hagen.de

Abstract

One of the key research issues with moving objects databases (MOD) is the modeling of moving objects. In this paper, a new moving objects database model, Moving Objects on Dynamic Transportation Networks (MODTN), is proposed. In MODTN, moving objects are modeled as moving graph points which move only within predefined transportation networks. To express general events of the system, such as traffic jams, temporary constructions, insertion and deletion of junctions or routes, the underlying transportation networks are modeled as dynamic graphs so that the state and the topology of the graph system at any time instant can be tracked and queried. Besides, to track the location of network constrained moving objects, a location update mechanism is provided, and the corresponding uncertainty management issues are analyzed.

1. Introduction

With the development of wireless communications and positioning technologies, the concept of moving objects databases (MOD) has become increasingly important, and has posed a great challenge to the database community. Existing database management systems (DBMS's) are not well equipped to handle continuously changing data, such as the locations of moving objects [19]. Therefore, new modeling methods are needed to solve this problem.

In recent years, a lot of research has been focused on the MOD technology, and many models and algorithms have been proposed. In [18, 19, 13], Wolfson *et al.* have proposed a Moving Objects Spatio-Temporal (MOST) model which is capable of tracking not only the current, but also the near future positions of moving objects. Su *et al.* in [15] have presented a data model for moving objects based on linear constraint databases. Chon *et al.* in [1] have proposed a Space-Time Grid Storage model for moving objects. In [7, 3, 8], Güting *et al.* have presented a data model and data structures for moving objects based on abstract data types. Besides, Pfoser and Jensen *et al.* in [10, 12] have discussed the indexing problem for moving object trajectories. However, nearly none of these works have treated the interaction between moving objects and the underlying transportation networks in any way.

More recently, increasing research interests are focused on modeling transportation networks and network constrained moving objects. Vazirgiannis *et al.* in [17] have discussed moving objects on fixed road networks, Papadias *et al.* in [9] have presented a framework to support spatial network databases. In [14], the authors have presented a computational data model for network constrained moving objects. Besides, the index problems of network constrained moving objects have also been studied [4, 11]. However, all these works have only considered static transportation networks, and moreover, none of these works have dealt with the relationship between the discrete presentation of moving objects and the location update policies. The uncertainty management issues for network constrained moving objects are not discussed either.

To explore these research issues, we propose a new moving objects database model, Moving Objects on Dynamic Transportation Networks (MODTN), in this paper. MODTN is an extension to our work presented in [6], which deals with the modeling of moving objects in static transportation networks at the abstract level.

The remaining part of this paper is organized as follows. Section 2 formally defines the MODTN model, Section 3 discusses the location update mechanism of MODTN, Section 4 analyzes location computation methods and uncertainty management issues, and Section 5 finally concludes the paper.

2. The MODTN Model

The modeling of moving objects on dynamic transportation networks is composed of two relatively independent steps. The first step is the modeling of the underlying transportation networks. Since the transportation networks can be subject to discrete changes over time, they should be modeled as "dynamic" graphs which allow us to express state changes (such as traffic jams and blockages caused by temporary constructions) and topology changes (such as insertion and deletion of junctions or routes). For simplicity, "dynamic

transportation networks" and "dynamic graphs" will be used interchangeably throughout this paper.

In modeling dynamic graphs, we utilize a state-based method. The basic idea is to associate a temporal attribute to every route or junction of the graph system so that the state of the route or junction at any time instant can be retrieved. Since the changes to the graph system are discrete, we can use a series of temporal units to represent a temporal attribute with each temporal unit describing one single state of the route or junction during a certain time period. As a result, the whole spatio-temporal history of the graph system can be stored and queried.

The second step is the modeling of moving objects on the graph system which has been handled in the first step. Since in most cases a moving object can be viewed as a point, moving objects are modeled as moving graph points in MODTN. A moving graph point is a function from time to graph point, which can be represented as a group of moving units in the discrete model. The methodology proposed in this paper can be easily extended to deal with more complicated situations where moving objects need to be modeled as moving graph lines or moving graph regions.

2.1. Overview of the System Architecture

In the following discussion, we suppose that in the whole MOD system, there can be multiple graphs coexisting while each graph is composed of a set of routes and a set of junctions. For each route, its geometry is described by a polyline so that it can actually assume an arbitrary shape instead of just a straight line. A junction connects two or more routes of the graph system. The connected routes can come from one graph (in this case, the junction is called "in-graph junction"), or belong to different graphs (in this case, the junction is called "inter-graph junction"). A junction can locate in the middle of a route, or at the beginning/end of the route. Figure 1 illustrates a graph system which is composed of two graphs.

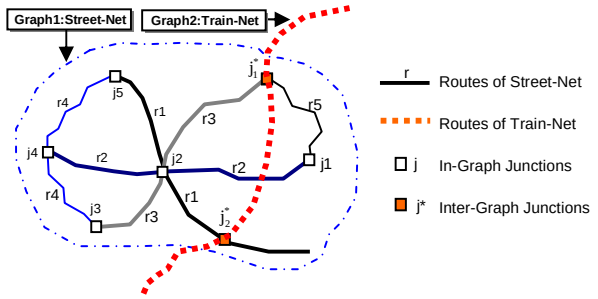


Figure 1. Graph system consisting of two graphs

Moving objects can move inside one graph and transfer from one route to another via in-graph junctions. They can also transfer from one graph to another via inter-graph junctions. In the system, both moving objects and the underlying transportation networks are dynamic – moving objects change their locations continuously, while transportation networks change their states and topologies discretely.

In order to envisage the above ideas, we give an example. This example shows how a modern logistic system works. We suppose that in such a system, transportation vehicles are uniquely identified and each of them is equipped with a portable computing platform and some other integrated location tracking equipments so that its location at any time instant can be retrieved.

We assume that such a logistic system, which is responsible for cargo delivery services, exists in the Verkehrsverbund Rhein-Ruhr (VRR) area of Germany. The whole highway network of the VRR area is expressed as a graph in the database. Besides, the street network of each city in this area is also stored as an independent graph. As a result, the whole system is composed of multiple graphs which can overlap each other, as shown in Figure 2.

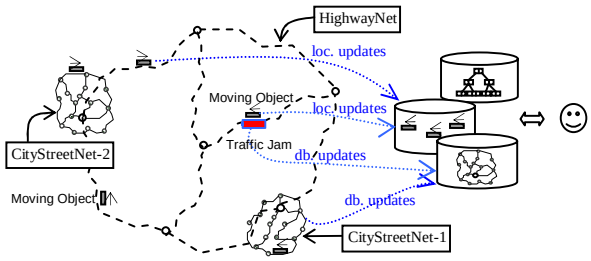


Figure 2. Architecture of the MODTN system

For a certain vehicle, it can move either by highway between two cities, or by street inside a city, during its whole journey. Therefore, it can pass through several different graphs during one trip. Since both historical and current location information is kept in the database, the system can support the following queries: "tell me the location of vehicle x310 at 2:00 PM of last Friday" and "find all vehicles that are currently in the Hagener street".

Besides, since the general events (such as traffic jams, car accidents, insertion and deletion of routes or junctions) of the graph system are stored in dynamic graphs, the following queries: "tell me the topology of the Hagen street network at time *t*" and "find the current traffic jam in the Hagener street and the moving objects affected by it" can also be handled.

To speed up query processing, both moving objects and dynamic graphs should be indexed. The database records and the index structures contain location information covering a time period from the past until the future. Therefore, when location updates occur, both database records and the index need to be modified.

2.2. The Data Model

Let's first deal with the transportation networks. In the MODTN model, the whole transportation networks are modeled as a dynamic graph system.

Definition 1 (dynamic graph system) A dynamic graph system, *GS*, is defined as a set of dynamic graphs and inter-graph junctions:

$$GS = \{G_1, G_2, \ldots, G_n, j_1^*, j_2^*, \ldots, j_m^*\}$$

where $n \geq 1$, $m \geq 0$, G_i $(1 \leq i \leq n)$ is a dynamic graph, and j_k^* $(1 \leq k \leq m)$ is an inter-graph junction (see Definition 5).

Definition 2 (dynamic graph) A dynamic graph, *G*, is defined as a pair:

$$G = (R, J)$$

where *R* is a set of dynamic routes and *J* is a set of dynamic in-graph junctions.

Definition 3 (dynamic route) A dynamic route of graph *G*, denote by *r*, is defined as follows:

$$r = (gid, rid, route, len, fdr, tp)$$

where *gid* and *rid* are identifiers of *G* and *r* respectively, *route* is a polyline which describes the geometry of *r*, *len* is the length of the route, $fdr \in \{0, 1, 2\}$ is the traffic flow directions allowed in the route, and *tp* is the temporal attribute (see Definition 6) associated with *r*.

The polyline *route* in the above definition can be defined as a series of points in the Euclidean space. For simplicity, we suppose that the graph system is spatially embedded in the X×Y plane so that the polyline can be presented as a series of points in the X×Y plane. The polyline is considered directed, whose direction is from the first vertex to the last vertex, which enables us to speak of the beginning point (or 0-end) and the end point (or 1-end) of the route.

The traffic flow directions allowed in a route can have three possibilities, which are specified by *fdr*, whose value can assume 0, 1, 2, which corresponds to "from 0-end to 1-end", "from 1-end to 0-end", and "both directions allowed" respectively.

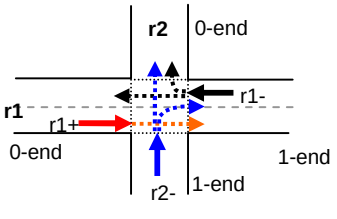
Definition 4 (dynamic in-graph junction) A dynamic in-graph junction of graph *G*, denoted by *j*, is defined as follows:

$$j = (gid, jid, loc, ((rid_i, pos_i))_{i=1}^{n}, m, tp)$$

where *gid* and *jid* are identifiers of *G* and *j* respectively, *loc* is the location of *j* which can be presented as a point value in the X×Y plane, $((rid_i, pos_i))_{i=1}^{n}$ describes the routes connected by *j*, *m* is the connectivity matrix of *j*, and *tp* is the temporal attribute associated with *j*.

(rid_i, pos_i) $(1 \leq i \leq n)$ in the above definition indicates the *i*th route connected by *j*, where rid_i is the identifier of the route and $pos_i \in [0, 1]$ describes the position of the junction inside the route. We suppose that the total length of any route is 1, and then every location in the route can be presented by a real number $p \in [0, 1]$.

The matrix *m* describes the connectivity of the junction. It contains possible matches of traffic flows in the routes connected by the junction, and the element value associated with each match can assume either 0 or 1, which indicates whether moving objects can transfer from the "in" traffic flow to the "out" traffic flow through this junction, as shown in Figure 3.



a) a junction and the traffic flows in it b) the connectivity matrix

Figure 3. A junction and its connectivity matrix

As illustrated in Figure 3, route *r*1 allows moving objects running in both directions so that it can have two traffic flows, r1+ and r1-. Route *r*2 is a one-way street so that it has only one traffic flow, *r*2-. These three traffic flows can have 9 combinations, and the value corresponding to each combination describes the transferability of the two traffic flows.

Definition 5 (dynamic inter-graph junction) A dynamic inter-graph junction, denoted by *j**, is defined as follows:

$$j^* = (jid, loc, tp, ((gid_i, rid_i, pos_i))_{i=1}^{n}, m, tp)$$

The definition of the inter-graph junction is very similar to that of the in-graph junction. The 3-tuple (gid_i, rid_i, pos_i) $(1 \leq i \leq n)$ describes the routes connected by *j**, which can come from different graphs.

Definition 6 (temporal attribute) The temporal attribute associated with a junction or a route, denoted by *tp*, describes the state history of the junction or route, which is defined as a sequence of the following form:

$$tp = ((I_i, s_i))_{i=1}^{n}$$

where I_i is a time interval, s_i is the state (see Definition 7) of the junction or route during I_i. (I_i, s_i) $(1 \leq i \leq n)$ is called the *i*th temporal unit of *tp*. For a valid temporal attribute, the following conditions should be satisfied:

(1) $\forall i, j \in \{1, \ldots n\}, i \neq j: I_i \cap I_j = \varnothing$

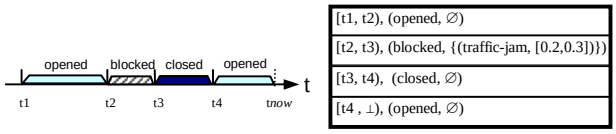
(2) $\forall i \in \{1, \ldots n\text{-}1\}: I_i \triangleleft I_{i+1}$ ($\triangleleft$ means "before" in time series)

(3) $\bigcup_{i=1}^{n} I_i = [\min(I_1), \max(I_n)]$.

For a certain temporal unit (I_i, s_i) $(1 \leq i \leq n)$, I_i is composed of two time instant values, $\min(I_i)$ and $\max(I_i)$,

which indicate the starting point and the endpoint of I_i respectively. $\min(I_i)$ must be a defined value while $\max(I_i)$ can be either defined or undefined. If $\max(I_i)$ is an "undefined" value $\perp$, then I_i is called an open temporal unit. Otherwise it is called a closed temporal unit. Semantically, $\perp$ means "until now". Therefore, if a junction or route is still active in the transportation network, its temporal attribute will contain exactly one open temporal unit, which forms its last temporal unit. Otherwise, if it has already been deleted from the transportation network, then its temporal attribute will only contain closed temporal units.

The insertion and deletion time of a junction or a route can be decided by $\min(I_1)$ and $\max(I_n)$ respectively. Figure 4 illustrates an example temporal attribute value.



a) state changes of a route b) the corresponding temporal units

Figure 4. An example temporal attribute value

Definition 7 (state) A state of a junction or a route, denoted by *s*, is defined as follows:

$$s = (\sigma, (br_i, BP_i)_{i=1}^{n})$$

where $\sigma \in$ {opened, closed, blocked}. If σ = blocked, then *s* must be associated with a route, and $(br_i, BP_i)_{i=1}^{n}$ is needed in this situation to describes the blockages of the route where br_i describes the reason (traffic-jam, construction, traffic-control, etc.), and $BP_i \subseteq [0, 1]$ describes the location, of the *i*th blockage of the route.

In the above definition, we assume that the location of the blockage is static so that it can be expressed as a closed interval over [0, 1], whose boundaries indicate the location of the borders of the blockage.

In dynamic transportation networks, a junction can have two states: opened and closed, and a route can have three states: opened, closed, and blocked. If a junction or a route is opened, then it is entirely available to moving objects. If a junction or a route is closed, then it is entirely unavailable to moving objects, which means that no moving objects are allowed to stay or move on any part of it. A closed junction or route is not deleted from the system. Instead, it is only temporarily unavailable to moving objects and can be reopened afterwards.

The blocked state is used to describe a special kind of state of a route, which means "partially available" to moving objects. That is, the unblocked part of the route is still available to moving objects, but no moving objects can move through the blocked part. Figure 5 gives an example blocked route.

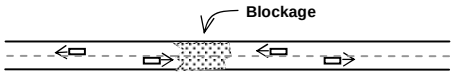


Figure 5. A blocked route with moving objects

In the dynamic graph system, since every junction or route has a temporal attribute associated, we can know its state at any given time instant. This is very useful in moving objects databases since a lot of queries can only be processed efficiently by accessing the states of the transportation networks. For instance, "please tell me all the routes which are currently blocked by traffic jams and the moving objects affected by them". Besides, through the temporal attribute, we can also know the life span of any junction or route of the graph system so that the topology changes of the transportation networks can also be expressed and queried. For instance, "find the shortest path from *a* to *b* at time instant *t*".

Based on the above definitions for dynamic transportation networks, we can then define some useful data types, graph point, graph route section, graph line, and graph region, which form the basis for the modeling and querying of moving objects. Let graph(*gid*), junct(*jid*), junct(*gid*, *jid*), route(*gid*, *rid*) be functions which return the graph, the junction, and the route corresponding to the specified identifiers respectively.

Definition 8 (graph point) A graph point is a point residing in the graph system. The set of graph points of graph system *GS*, denoted by *GP*, is defined as follows:

$$GP = \{jid^* \mid \text{junct}(jid^*) \in \text{interjuncts}(GS)\} \cup \{(gid, jid) \mid \text{junct}(gid, jid) \in \text{injuncts}(GS)\} \cup \{(gid, rid, pos) \mid \text{route}(gid, rid) \in \text{routes}(GS), pos \in [0,1]\}$$

where interjuncts(*GS*), injuncts(*GS*) and routes(*GS*) are the set of inter-graph junctions, the set of in-graph junctions, and the set of routes of *GS* respectively.

Definition 9 (graph route section) A graph route section is a part of a route. The set of graph route sections of graph system *GS*, denoted by *GRS*, can be defined as follows:

$$GRS = \{(gid, rid, S) \mid \text{route}(gid, rid) \in \text{routes}(GS), S \subseteq [0,1]\}$$

Definition 10 (graph line) A graph line is a polyline inside the graph system, which can be defined by just specifying the vertices of the polyline. The set of graph lines of graph system *GS*, denoted by *GL*, can be defined as the following form:

$$GL = \{ (vertex_i)_{i=1}^{n} \mid n \geq 2, vertex_i = (gid_i, rid_i, pos_i) \in GP,$$

and:

(1) $\forall i \in \{1, \ldots n\text{-}1\}$:
$\text{route}(gid_i, rid_i) = \text{route}(gid_{i+1}, rid_{i+1}) \vee$
$\text{Eucl}(vertex_i) = \text{Eucl}(vertex_{i+1}) = \text{Eucl}(\text{getjunct}(vertex_i, vertex_{i+1}))$;

(2) $\forall i \in \{1, \ldots n\text{-}1\} : \text{viable}(vertex_i, vertex_{i+1}) \}$

In the above definition, the function Eucl(*gp*) returns the Euclidean value of a graph point, getjunct($vertex_i$, $vertex_{i+1}$) returns the junction in which $vertex_i$, $vertex_{i+1}$ are located. viable($vertex_i$, $vertex_{i+1}$) means that through route(gid_i, rid_i) or getjunct($vertex_i$, $vertex_{i+1}$), moving objects can transfer from $vertex_i$ to $vertex_{i+1}$.

Graph lines are considered as directed paths in the transportation network, whose directions are determined by the vertex series.

Definition 11 (graph region) A graph region is defined as a set of junctions and a set of route sections. The set of graph regions of the graph system *GS*, denoted by *GR*, is defined as follows:

$$GR = \{(V, W) \mid V \subseteq GJ, W \subseteq GRS\}$$

where $GJ = \{jid^* \mid \text{junct}(jid^*) \in \text{interjuncts}(GS)\} \cup \{(gid, jid) \mid \text{junct}(gid, jid) \in \text{injuncts}(GS)\}$. Different from the graph line, a graph region can contain an arbitrary set of graph route sections.

Based on the above definitions of network constrained data types, we can then model moving objects on the dynamic graph system. In MODTN, moving objects are modeled as moving graph points.

Definition 12 (moving graph point) A moving graph point, *mgp*, is defined as a function from time to graph point, that is:

$$mgp = f: T \rightarrow GP$$

where *T* is the domain of time, and *GP* is the domain of graph point of the graph system.

In the above definitions, most data types are defined discretely so that they can be implemented directly. The only exception is the moving graph point data type. In implementation, Definition 12 should be translated into a discrete representation. That is, a moving graph point is expressed as a set of moving units, and each moving unit describes one single moving pattern of the moving object for a certain period of time.

Definition 13 (discrete presentation of moving graph point) a discrete presentation of moving graph point, *dmgp*, is defined as a sequence:

$$dmgp = ((t_i, (gid_i, rid_i, pos_i), vm_i))_{i=1}^{n}$$

where t_i is a time instant, $(gid_i, rid_i, pos_i) = gp_i$ is a graph point describing the location of the moving object at time t_i, and vm_i is the speed measure of the moving object at time t_i. $(t_i, (gid_i, rid_i, pos_i), vm_i) = \mu_i$ is called the *i*th moving unit of *dmgp*.

The speed measure *vm* is a real number value. Its abstract value is equal to the speed of the moving object, while its sign (either positive or negative) depends on the direction of the moving object. If the moving object is moving from 0-end towards 1-end, then the sign is positive. Otherwise, if it is moving from 1-end to 0-end, the sign is negative.

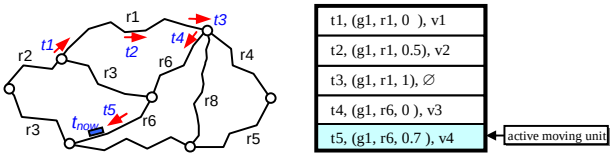
For a valid discrete presentation of moving graph point, the following conditions should be met:

(1) $\forall i \in \{1, \ldots n\text{-}1\}$:
route(gid_i, rid_i) = route(gid_{i+1}, rid_{i+1}) $\vee$
Eucl(gp_i) = Eucl(gp_{i+1}) = Eucl(getjunct (gp_i,gp_{i+1}));

(2) $\forall i \in \{1, \ldots n\text{-}1\}$: viable(gp_i, gp_{i+1});

(3) $\forall i \in \{1, \ldots n\text{-}1\}$: $t_i < t_{i+1}$, and the moving object is assumed to move at even speed between t_i and t_{i+1}.

Figure 6 gives an example discrete representation of a moving graph point.



a) journey of a moving object b) corresponding moving unites

Figure 6. An example moving graph point value

In Definition 13, we assume that between two consecutive moving units, moving objects move at even speed. As a result, the location of moving objects can be computed by interpolation (see Subsection 4.1). However, this is only an ideal situation. In real world MOD applications, the moving units are generated by location updates (see Section 3), and the moving object is only moving roughly at even speed between two moving units. As a result, the location of moving objects is uncertain between two location updates, and we have to take the uncertainty problem into consideration. When uncertain management is considered, the above definition of moving point is actually interpreted as a "moving route section". In this paper, we still call this definition "moving graph point" to keep consistency.

For a running moving object, its last moving unit contains predicted information. We call the last moving unit "active moving unit", which contains key information for location update strategies (see Section 3).

3. Location Update Strategies of MODTN

The above data model enables us to present the information of moving objects and the underlying transportation networks in databases. However, this is still not enough because for a running MOD system, it is impossible to get all the real-time information from the system manually so that a mechanism should be provided to collect the data automatically.

For the transportation networks, since the topology and state changes are discrete and relatively less frequent, the database records can be maintained manually by issuing update commands. It is also possible that some of the state changes (such as traffic jams) can be reported to the database automatically. For instance, some techniques in

transportation engineering enable the system to detect car accidents and their locations automatically by examining the change of traffic flows so that the database can be updated real-time to reflect up-to-date situations.

However, it is nearly impossible to track the real-time location of moving objects in a running MOD system by manually issuing database commands. Therefore, we need a location update mechanism to track moving objects automatically.

In [18, 19] Wolfson *et al*. have discussed the location update policies for the MOST model. The basic idea can be summarized as "predict and compare". That is, the system makes a prediction according to the current moving pattern of the moving object. During its move, the moving object compares its actual position measured by GPS with the computed position. Whenever the difference between them reaches a certain threshold, a location update is triggered to modify the database information. This idea provides a general principle for location update policies in MOD system. In MODTN model, however, this basic principle can be optimized since the moving objects are not modeled in the Euclidean space directly, but modeled in predefined transportation networks.

3.1. Basic Ideas of Location Update in MODTN

In MODTN, we suppose that every moving object is equipped with a portable computing platform which is connected with some other integrated devices such as the sensor communicator, the wireless interface, and the milemeter which measures the distance covered by the moving object in a certain route. In every junction of the transportation networks, there are a group of sensors installed so that whenever a moving object transfers from one route to another via the junction, it gets a notification which will trigger a location update. The sensors are associated with some information exchanging equipments which will send the route identifier, the location of the junction inside the route, and the sign for speed measures to the moving object when it enters into a new route. (As an alternative to the sensors and milemeters, GPS can also be used to fulfill the location update purpose. With a local algorithm, the moving object can transform the location information from the GPS (with the (*x*, *y*) format) to the (*rid*, *pos*) form, where *rid* is the identifier of the route where the moving object is located, and pos $\in$ [0, 1] is the location of the moving object inside the route. In this paper, we focus on the sensor alternative. The same location update strategies are suited for the GPS case).

The basic idea behind the location update policy of MODTN is the "Inertia Principle". That is, the system assumes that the moving object will continue to move along the current route at roughly steady speed for some more time, and whenever this assumption becomes invalid, the moving object will initiate a location update so that the up-to-date information of the moving object can be reported to the database server.

When a moving object *mo* initiates its journey in the MOD system from a junction, it needs to send a location update message *msgu* to the server, which contains the following information:

$$msgu = (mid, t_u, (gid_u, rid_u, pos_u), vm_u)$$

where *mid* is the identifier of *mo*, t_u is the time when the location update is triggered, $(gid_u, rid_u, pos_u) = gp_u$ is the location (expressed as a graph point) of *mo* at time t_u, and vm_u is the speed measure of *mo* at time t_u. The sign of vm_u can be determined from the information received from the sensor when the moving object enters into a new route via a junction.

When receiving this first location update message, the server will extract the information contained in it and generate a corresponding moving unit. This moving unit will be saved to the moving graph point value of the moving object (at this moment, the moving unit is also the active moving unit). The moving object also needs to keep the active moving unit for location update purposes.

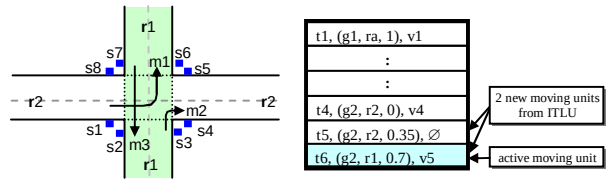
During its move, the moving object will compare its actual moving parameters (current route identifier, location, and speed) with the moving pattern contained in the active moving unit. Whenever certain conditions are met, a location update will be triggered so that the location of the moving object can be tracked. In MODTN, there are 3 kinds of location updates, the ID-Triggered Locations Update (ITLU), the Distance-Threshold-Triggered Location Update (DTTLU), and the Speed-Threshold-Triggered Location Update (STTLU). Among them, only ITLU and DTTLU are basic ones while STTLU is optional and is needed only when uncertainty management is involved (see Subsection 4.2).

When receiving a location update message, the server will extract the information contained in the message and generate a corresponding moving unit. This new unit is then appended to the corresponding moving graph point value of the moving object.

3.2. ID-Triggered Location Update (ITLU)

As stated earlier, in MODTN, every junction is equipped with a group of sensors so that whenever a moving object transfers from one route to another, a location update will be triggered to reflect the change of route identifiers, as shown in Figure 7 (we draw the sensors according to their functionalities. In real-world applications some of them can be combined).

In Figure 7(a), moving objects *m*1 and *m*2 will trigger an ITLU respectively, while *m*3 will not, since it doesn't change to another route even though it passes through a junction.



a) ID-triggered location update b) corresponding moving unites of m1

Figure 7. ID-triggered location update

For *m*1, suppose that it passes by sensor *s*1 at time *t*5, and passes by sensor *s*6 at time *t*6. The location update will be triggered at time *t*6, and two location update messages will be sent to the server via one communication package through the wireless interface. The server can then extract two moving units and save them to the corresponding moving graph point value of *m*1, as shown in Figure 7(b).

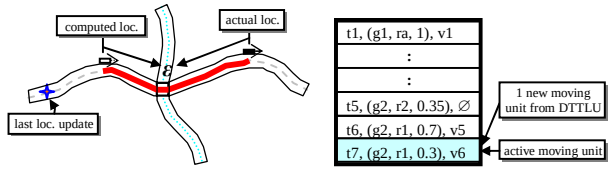
At time *t*6, except the location update, some other computations and adjustments are also need for moving object *m*1. For instance, the information exchanging equipment associated with sensor *s*6 will send the identifier of the new route, the location of the junction inside the new route, and the sign of speed measures to *m*1. Besides, the milemeter of *m*1 is refreshed according to the newly received information.

3.3. Distance-Threshold-Triggered Location Update (DTTLU)

During its move along a certain route, *mo* will compare its actual position measured by the milemeter with the computed position derived from the active moving unit. The computed position at the current time instant t_{now}, denoted by pos_{now}, can be computed with the following formula:

$$pos_{now} = pos_n + vm_n \times (t_{now} - t_n)$$

where pos_n and t_n are the position and the time corresponding to the last location update respectively. If the difference between the actual position and pos_{now}, denoted by ε, exceeds a certain predefined threshold ξ (for instance, 0.5 kilometer) then a new location update is triggered to report the actual location of the moving object, as shown in Figure 8.



a) distance-threshold-triggered loc. update b) corresponding moving unites

Figure 8. Dist.-threshold-triggered loc. update

During this process, some transformation is needed since the speed and the milemeter are using real measuring unites (such as kilometer) while the graph positions are expressed with a real number value $p \in [0, 1]$. However, this transformation is trivial since the length of every route is available from the database.

When evaluating the computed position, a special case should be considered when the moving object is near the end of the route and the actual speed is lower than the predicted one. In this case, the computed position can exceed the scope of [0, 1] and we can interpret these extra values as the distance covered by the moving object in other routes after the current route is finished, so that the location update policy does not need to be changed.

When receiving a distance threshold triggered location update message, the server will extract one moving unit from it and will append the unit to the corresponding moving graph point value of the moving object, as illustrated in Figure 8(b).

3.4. Speed-Threshold-Triggered Location Update (STTLU)

If we compute the locations of moving objects through interpolation (see Subsection 4.1), ITLU and DTTLU are enough to track the locations of moving objects. However, for the sake of uncertainty management (see Subsection 4.2), especially to reduce uncertainty as more as possible, we need another kind of location update, Speed-Threshold-Triggered Location Update (STTLU).

As stated earlier, the active moving unit of the moving object, μ_n, contains the current moving pattern of the moving object, and the moving object is expected to move roughly at the speed indicated in μ_n. During its move, the moving object will compare its actual speed with the speed contained in μ_n. Whenever the difference between them exceeds a certain predefined threshold ψ (for instance, 10 kilometer/hour), then a location update is triggered. In this way, we can be assured that between any two consecutive location updates (suppose the corresponding moving units are $\mu_i = (t_i, (gid_i, rid_i, pos_i), vm_i)$ and $\mu_{i+1} = (t_{i+1}, (gid_{i+1}, rid_{i+1}, pos_{i+1}), vm_{i+1})$, the speed of the moving object is between $(|vm_i| - \psi)$ and $(|vm_i| + \psi)$ ($|vm_i|$ is the abstract value of vm_i).

When receiving a speed-threshold-triggered location update message, the server will extract one moving unit from it and append the new moving unit to the corresponding moving graph point value of the moving object.

4. Querying the Location of Moving Objects

In this section, we discuss how the location of a moving object can be computed from its moving units. We suppose that the corresponding moving graph point value of an active moving object, *mo*, is as follows:

$mgpoint = ((t_i, (gid_i, rid_i, pos_i), vm_i))_{i=1}^{n}$

Let $\mu_i = (t_i, (gid_i, rid_i, pos_i), vm_i)$ $(1 \le i \le n)$ be the ith moving unit of the moving object, and $gp_i=(gid_i, rid_i, pos_i)$ be the location of the moving object at time t_i. For the sake of simplicity, we assume that the speed measure vm_i is positive, which means that the moving object is moving from 0-end towards 1-end along route(gid_i, rid_i). The methodology can be easily adapted to the situation when the speed measure is negative.

Let $v^{i}_{\max} = vm_i + \psi$ and $v^{i}_{\min} = vm_i - \psi$ where ψ is the speed threshold.

4.1. Computing the Locations of Moving Objects through Interpolation

The location of a moving object can be computed through interpolation. By interpolation, the move of the moving object between any two location updates is approximated to an even speed move. For the query: "where is moving object *mo* at time t_q?", the answer, which is a graph point $gp_q = (gid_q, rid_q, pos_q)$, can be computed in the following way.

Case 1. $\exists\, i \in \{1, \ldots, n\} : t_q = t_i$

In this case, t_q happens to be a location update time, and the location information contained in μ_i can be returned directly as the result. That is:
$gp_q = (gid_i, rid_i, pos_i)$

Case 2. $\exists\, i \in \{1, \ldots, n-1\} : t_i < t_q < t_{i+1}$

In this case, t_q is between two consecutive location updates, and the corresponding moving units μ_i, μ_{i+1} need to be further checked in this situation. If route(gid_i, rid_i) = route(gid_{i+1}, rid_{i+1}), then according to the location update policies described in Section 3, we can be assured that the moving object is on route(gid_i, rid_i) at time t_q, and its location at t_q is a graph point $gp_q = (gid_i, rid_i, pos_q)$ where pos_q can be computed with the following formula:

$$pos_q = pos_i + \frac{pos_{i+1} - pos_i}{t_{i+1} - t_i} \times (t_q - t_i)$$

If route(gid_i, rid_i) ≠ route(gid_{i+1}, rid_{i+1}), then from the location update strategies described in Section 3 we know that μi and μi_{+1} are generated by an ITLU, Eucl(gp_i) = Eucl(gp_{i+1}), and at time t_q the moving object is in junction getjunct(gp_i, gp_{i+1}). The graph point corresponding to this junction will be returned as the final result.

Case 3. $t_n < t_q \le t_{now}$

In this case, we know that the moving object is still on route(gid_n, rid_n). Otherwise, there would be an ITLU triggered after t_n. Therefore, the location of the moving object at time t_q is a graph point $gp_q = (gid_n, rid_n, pos_q)$ where pos_q can be computed as follows:

$$pos_q = pos_n + v_n \times (t_q - t_n)$$

By computing the location of a moving object through interpolation, the location of the moving object at any time instant can be simply presented as a graph point. As a result, the query processing mechanism and the query language of the MOD system can be simplified. Besides, the location update mechanism can also be simplified since the third kind of location update, STTLU, is not necessary in this case.

The result from the above computing method is only an approximate description of the actual location, and the error introduced is closely related to the distance threshold ξ.

4.2. Querying Moving Objects with Uncertainty Considered

Even though in a lot of MOD applications, the interpolation technique described in Subsection 4.1 is sufficient, a better solution is to take the uncertainty brought about by the location update policy into consideration. As stated in [19], the location of a moving object other than location update time is actually uncertain. Therefore, we should introduce the concept of "possible location" in presenting the location of the moving object instead of just expressing it as a precise point.

In MODTN, the uncertainty management problem can be better solved because the possible location of a moving object at any historical or present time instant is reduced to a route section (See Figure 9). In the following discussion, we suppose that ξ, $v^{i}_{\max}$, and $v^{i}_{\min}$ have already been transformed to the [0, 1] scope according to the length of the corresponding route. Besides, we will focus on the uncertainty caused by sampling method alone so that we assume the uncertainty caused by other factors to be negligible.

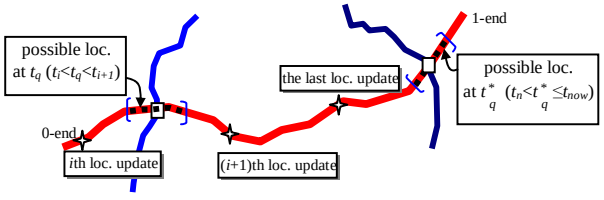


Figure 9. Possible locations of a moving object

Case 1. $\exists\, i \in \{1, \ldots, n\} : t_q = t_i$

In this case, the possible location of the moving object is a graph point $gp_q = (gid_i, rid_i, pos_i)$.

Case 2. $\exists\, i \in \{1, \ldots, n-1\} : t_i < t_q < t_{i+1}$

If route(gid_i, rid_i) = route(gid_{i+1}, rid_{i+1}), then we can be assured that the moving object is on route(gid_i, rid_i), and

its possible position is a graph route section grs_q = (gid_i, rid_i, seg_q) where $seg_q \subseteq [0, 1]$ and satisfies the following conditions:

1) $seg_q \subseteq [\ pos_q^* - \xi,\ pos_q^* + \xi]$, where $pos_q^* = pos_i + vm_i \times (t_q - t_i)$. Otherwise there would be a DTTLU between t_i and t_{i+1};

2) $seg_q \subseteq [pos_i + v^i_{\min} \times (t_q - t_i),\ pos_i + v^i_{\max} \times (t_q - t_i)]$, where $v^i_{\min} \times (t_q - t_i)$ and $v^i_{\max} \times (t_q - t_i)$ are the shortest and the longest distances the moving object can cover during $\triangle t$ ($\triangle t = t_q - t_i$) time without triggering an STTLU;

3) $seg_q \subseteq [pos_{i+1} - v^i_{\max} \times (t_{i+1} - t_q),\ pos_{i+1} - v^i_{\min} \times (t_{i+1} - t_q)]$. Otherwise, the moving object would not be able to arrive at gp_{i+1} in time without triggering a speed triggered location update.

To sum up, the possible location of the moving object is:

$$seg_q = [0, 1] \cup [\ pos_q^* - \xi,\ pos_q^* + \xi] \cup [pos_i + v^i_{\min} \times (t_q - t_i),\ pos_i + v^i_{\max} \times (t_q - t_i)] \cup [pos_{i+1} - v^i_{\max} \times (t_{i+1} - t_q),\ pos_{i+1} - v^i_{\min} \times (t_{i+1} - t_q)]$$

where $pos_q^* = pos_i + vm_i \times (t_q - t_i)$.

If route(gid_i, rid_i) ≠ route(gid_{i+1}, rid_{i+1}), then we know that μ_i and μ_{i+1} are generated by an ITLU. In this case, the possible location of the moving object is a graph point which corresponds to getjunct(gp_i, gp_{i+1}).

Case 3. $t_n < t_q \le t_{now}$

In this case, we know that the moving object is still on route(gid_n, rid_n). Otherwise, there would be an ITLU triggered after the last location update. Therefore, the location of the moving object at time t_q is a graph route section grs_q = (gid_n, rid_n, seg_q) where $seg_q \subseteq [0, 1]$ and satisfies the following conditions:

1) $seg_q \subseteq [\ pos_q^{\Diamond} - \xi,\ pos_q^{\Diamond} + \xi]$, where $pos_q^{\Diamond} = pos_n + vm_n \times (t_q - t_n)$. Otherwise there will be a DTTLU triggered after t_n;

2) $seg_q \subseteq [pos_n + v^n_{\min} \times (t_q - t_n),\ pos_n + v^n_{\max} \times (t_q - t_n)]$.

Otherwise there will be an STTLU triggered after t_n; Therefore, seg_q can be computed as follows:

$$seg_q = [0, 1] \cup [\ pos_q^{\Diamond} - \xi,\ pos_q^{\Diamond} + \xi] \cup [pos_n + v^n_{\min} \times (t_q - t_n),\ pos_n + v^n_{\max} \times (t_q - t_n)]$$

where $pos_q^{\Diamond} = pos_n + vm_n \times (t_q - t_n)$.

By computing the location of the moving object with uncertainty involved, the moving graph point defined in Definition 13 is actually interpreted as a moving graph route section. We still call the definition "moving graph point" in this paper just for the sake of consistency. Besides, when uncertainty is involved, we need to adapt related operations to the uncertainty context. For instance, the **inside** operation can be extended to **inside_possibly** and **inside_definitely**, as stated in [16].

4.3. Prediction of the Future Locations of Moving Objects

Since we assume that moving objects can only move inside the predefined transportation networks, we can predict their future locations more accurately. Suppose the query is: "tell me the location of moving object *mo* at t_q ($t_q > t_{now}$)". The predicted location of the moving object can be computed in the following way (we assume that moving objects do not submit moving plans proactively).

First, the algorithm needs to search in the graph system and to decide a "foreseeable future path", *ffp*, according to the possible position of the moving object at time t_{now}(see Subsection 4.2). *ffp* is a graph line which starts from the computed position of the moving object at time t_{now}(see Subsection 3.3) and finishes at the first junction after which multiple consecutive traffic flows exist, as shown in Figure 10.

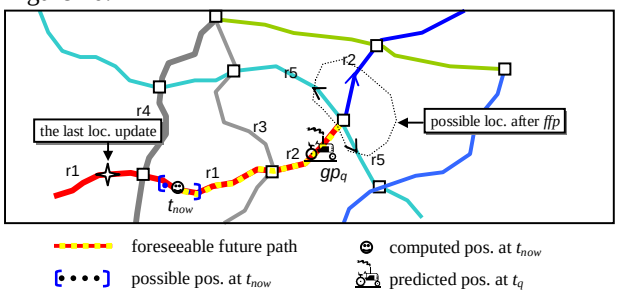


Figure 10. Predicting future locations

Then, the system needs to predict a future speed for the moving object. This can be fulfilled either by using the speed contained in μ_n (if t_q is not too far away from t_{now}), or by computing an average speed from the speed information contained in the moving units of the moving object. In the latter case, the speed can be computed like this:

$$\overline{v} = \frac{\sum_{i=1}^{n-1} (|vw_i| \times (t_{i+1} - t_i)) + |vw_n| \times (t_{now} - t_n)}{t_{now} - t_1}$$

where $|vm|$ is the abstract value of vm. With $\overline{v}$ known, we can predict the distance the moving object can cover in $\triangle t^*$ time ($\triangle t^* = t_q - t_{now}$) as follows:

$$d = \overline{v} \times \triangle t^*$$

From *ffp* and d we can get a graph point value gp_q which is the predicted position of the moving object at time t_q. According to different applications, gp_q can be either presented directly as the final result, or extended to a graph line value. In the latter case, we need to impose an

error-factor, which can be a function of $\triangle t^*$, to gp_q so that the final result can be a graph line value.

In MODTN we confine the prediction to the scope of the foreseeable future path *ffp*, since after *ffp* the moving object can have multiple possible directions, so that its possible position can explode, as shown in Figure 10.

5. Conclusions

One of the key research issues with moving objects databases (MOD) is the modeling of moving objects. In this paper, a new moving objects database model, Moving Objects on Dynamic Transportation Networks (MODTN), is proposed. In MODTN, transportation networks are modeled as dynamic graphs and moving objects are modeled as moving graph points. Besides, a location update mechanism is provided and the related uncertainty management issues are analyzed.

We have designed a rich set of data types and operations for moving objects and the underlying dynamic graphs (which will be presented in another paper). These data types and operations have been partly implemented in *C++* as three algebra modules, spatial algebra, dynamic graph algebra, and moving object algebra, in the Secondo system [2]. Secondo is a new generic environment supporting the implementation of database systems for a wide range of data models and query languages. Besides, a graphical user interface, which can display spatial objects, transportation networks, and moving objects, has been implemented in Java.

Compared with other moving object models, MODTN has the following features: 1) the system is enabled to support logic road names, while queries based on Euclidean space can also be supported; 2) both history and current location information can be queried, and the system can also support future location queries based on the predicted information; 3) location update policies can be optimized since the change of direction alone will not trigger a location update; 4) uncertainty problem can be better managed because the possible position of a moving object at any historical or present time instant is reduced to a moving route section; and 5) general events of the system, such as blockages and topology changes can also be expressed so that the system is enabled to deal with the interaction between the moving objects and the underlying transportation networks.

Acknowledgements

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) research project "Databases for Moving Objects" under the grant number Gu-293/8-1.

References

[1] Chon H D, Agrawal D, Abbadi A E. Using Space-Time Grid for Efficient Management of Moving Objects, *Proc. of MobiDE 2001*, CA, USA, 2001.

[2] Dieker S, Güting R H, Plug and Play with Query Algebras: SECONDO. A Generic DBMS Development Environment. *Proc. of IDEAS 2000*, Yokohoma, Japan, 2000.

[3] Forlizzi L, Güting R H, Nardelli E, Schneider M. A Data Model and Data Structures for Moving Objects Databases. *Proc. ACM SIGMOD Conference*, TX, USA, 2000.

[4] Frentzos E. Indexing objects moving on fixed networks, *Proc. of SSTD'03*, Santorini island, Greece, July, 2003.

[5] Güting R H, Second-Order Signature: A Tool for Specifying Data Models, Query Processing, and Optimization. *Proc. ACM SIGMOD Conference*. Washington, USA, 1993.

[6] Güting R H, Almeida V T, Ding Z, Modeling and Querying Moving Objects in Networks, Fernuniversität Hagen, Informatik-Report 308, 2004.

[7] Güting R H, Böhlen M H, Erwig M, Jensen C S, Lorentzos N A, Schneider M, Vazirgiannis M. A Foundation for Representing and Querying Moving Objects. *ACM Transactions on Database Systems*, 25(1), 2000.

[8] Lema J A C, Forlizzi L, Güting R H, Nardelli E, Schneider M, Algorithms for Moving Objects Databases. *The Computer Journal*, 46(6), 2003

[9] Papadias D, Zhang J, Mamoulis N, Tao Y. Query processing in spatial network databases, *Proc. of VLDB'03*, Berlin, Germany, 2003.

[10] Pfoser D, Jensen C S, Theodoridis Y. Novel Approach to the Indexing of Moving Object Trajectories. *Proc. of VLDB'00*, Cairo, Egypt, 2000.

[11] Pfoser D, Jensen C S. Indexing of Network-Constrained Moving Objects, *Proc. of GIS'03*, Louisiana, USA, 2003

[12] Saltenis S, Jensen C S, Leutenegger S T, Lopez M A. Indexing the Position of Continuously Moving Objects. *Proc. of ACM SIGMOD 2000*, TX, USA, 2000.

[13] Sistla A P, Wolfson O, Chamberlian S, Dao S. Modeling and querying Moving Objects. *Proc. of ICDE 1997*, Birmingham, UK, 1997.

[14] Speicys L, Jensen C S, Kligys A. Computational data modeling for network-constrained moving objects, *Proc. of GIS'03*, Louisiana, USA, 2003

[15] Su J, Xu H, Ibarra O. Moving Objects: Logical Relationships and Queries, *Proc. of SSTD'01*, CA, USA, 2001.

[16] Trajcevski G, Wolfson O, Chamberlain S, Zhang F, The Geometry of Uncertainty in Moving Objects Databases, *Proc. of EDBT'02*, Prague, Czech Republic, March 2002.

[17] Vazirgiannis M, Wolfson O. A Spatiotemporal Query Language for Moving Objects on Road Networks, *Proc. of SSTD'01*, CA, USA, 2001.

[18] Wolfson O, Chamberlain S, Dao S, Jiang L. Location Management in Moving Objects Databases. *Proc.of WOSBIS'97*, Budapest, Hungary, 1997.

[19] Wolfson O, Xu B, Chamberlain S, Jiang L. Moving Object Databases: Issues and Solutions. *Proc. of SSDBM'98*, Capri, Italy, July 1998.