Probabilistic Range Queries for Uncertain Trajectories on Road Networks

Kai Zheng[1] Goce Trajcevski[2] Xiaofang Zhou[1,3] Peter Scheuermann[2]

[1] School of ITEE, The University of Queensland, Australia

[2] Department of EECS, Northwestern University, USA

[3] School of Information, Renmin University of China, China
Key Lab of Data Engineering and Knowledge Engineering, Ministry of Education,China

kevinz@itee.uq.edu.au, goce@eecs.northwestern.edu, zxf@uq.edu.au, peters@eecs.northwestern.edu

ABSTRACT

Trajectories representing the motion of moving objects are typically obtained via *location sampling*, e.g. using GPS or road-side sensors, at *discrete time-instants*. In-between consecutive samples, nothing is known about the whereabouts of a given moving object. Various models have been proposed (e.g., sheared cylinders; space-time prisms) to represent the uncertainty of the moving objects both in unconstrained Euclidian space, as well as road networks. In this paper, we focus on representing the uncertainty of the objects moving along road networks as *time-dependent probability distribution functions*, assuming availability of a maximal speed on each road segment. For these settings, we introduce a novel *indexing mechanism* – UTH (Uncertain Trajectories Hierarchy), based upon which *efficient algorithms* for processing spatio-temporal range queries are proposed. We also present experimental results that demonstrate the benefits of our proposed methodologies.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithm, Performance

Keywords

Uncertain Trajectories, Probabilistic Range Queries, Road Networks

1. INTRODUCTION

With the fast advances in communication and sensing/positioning technologies, many important applications such as mobile communication systems, vehicle guidance and traffic management systems, geographical information systems, etc, rely on Moving Objects Databases (MOD) [13] to efficiently retrieve data and process queries.

EDBT 2011, March 22–24, 2011, Uppsala, Sweden.

Literature in this area can be roughly classified into two categories according to the settings they considered:

- The information arrives as a stream of location samples obtained, e.g., from the on-board GPS devices or roadside sensors. Research work in this settings focus on improving the efficiency of algorithms by avoiding unnecessary re-evaluation upon certain updates [20, 10].
- The information about moving objects *trajectories* is available as a sequence of *(location, time)* values, corresponding to the *past* motions – or, corresponding to the predicted future motions, obtained via some route-planning tools [8, 19].

In some sense, our work belongs to the second kind of settings – i.e., we assume that *location-in-time samples* of the moving objects are already available in the MOD. Traditionally, when querying the moving objects, a normal assumption is that the entire trajectories are available. However, at the heart of the motivation for this work, is the observation that the GPS devices and/or roadside sensors can only provide location samples at *discrete time-instants*, and nothing is known about the objects' whereabouts in-between two consecutive samples. To tackle this problem and query about the object's movement in-between those samples, a typical technique is to apply interpolation [19] by which means the sampled positions become the end points of line segments, and the trajectories are transformed into polylines in 3D (*x-y-time*) space. However, as pointed out in several work [4, 18, 23], interpolation cannot reflect the exact movement pattern of an object. In fact, a moving object can be located anywhere within a given (bounded) region, for as long as it does not violate certain constraints (e.g., maximum allowed velocity). In the light of this observation, various data models and query processing algorithms that incorporate *uncertainty* in the representation of trajectories and semantics of queries have been proposed [3, 4, 18, 23, 30, 29, 28].

While the above approaches target the moving objects in free space, recent years have brought about many results on spatio-temporal query processing in *Spatial Network Database* (SNDB) and road networks [6, 14, 16, 21, 22, 26], since in many practical scenarios objects move in a constrained environment (roads, railways, rivers, etc) rather than free space. Similar with the free space, moving objects on road networks are also observed at particular time instants, which means uncertainty inherently exists in-between those sampled positions. Consider a taxi travelling at the speed of 50km/h, if it sends its current location to central server every 3 minutes (which is quite normal in taxi management applications), any consecutive points of the trajectory record will be

about 2.5km away from each other. Even the movement of taxi is restricted on road segments, little is known about its exact location at specific time instant since the path connecting these two sampled locations is usually not unique given the distance is that long. Besides, travel speed can hardly be constant in normal traffic area. This kind of uncertainty will further increase when the sampling rate is even lower, which can occur in many scenarios, such as demand for energy and communication cost reduction [10, 33] in low battery equipments, non-availability of GPS coverage in certain areas, etc. Motivated by these observations, in this paper we will incorporate uncertainty into trajectories on road networks and support *continuous range queries* for them. However, we cannot apply the models and algorithms for uncertain trajectories in free space [23, 18, 30, 29, 28], since the uncertain region is entirely different when the movement is constrained in road networks. In the sequel, we use the following example to better explain the impact of uncertainty on the range query semantics.

1.1 Motivational Example

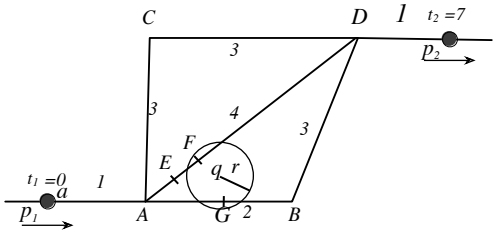


Figure 1: Range Query on Road Network

Assume that a given object a is moving along a road network, as illustrated in Figure 1, and that there have been two consecutive location-samples: p_1 at $t_1 = 0$ and p_2 at $t_2 = 7$. Each road segment has associated length and maximum speed, allowed by law or other physical constraints (e.g., road-work, weather/visibility, etc.) – from which its minimum travel time costs can be derived. In the scenario of Figure 1, these numbers are indicated along the respective edges and, as shown, there are three possible routes in-between the location samples, specifically, between the intersections A and D : – $(\overline{AC}, \overline{CD})$ with a travel time 3 + 3 = 6 time units; – $(\overline{AD})$ with a travel time of 4 units; – and $(\overline{AB}, \overline{BD})$ with a travel time of 2 + 3 = 5 time units. Now, consider the following query:
Q_R: *Retrieve all the moving objects that are within distance r from the location q at time $t = 2$.*

When it comes to the object a, the question becomes: *should a be returned as (part of) the answer to Q_R?*

A straightforward idea would be to attempt to construct a deterministic trajectory of a, e.g., via interpolation, and then solve the problem in a deterministic manner. The two key assumptions behind this are: (a) the object moves along a particular route, e.g., the optimal one in terms of travel time; and (b) the object travels with a constant speed. With these assumptions, we can infer that in-between the two samples, the object travelled along the route $\overline{p_1 AD p_2}$ and its location at $t = 2$ is E, which means, the object a is not part of the answer-set of Q_R. However, this solution is, in a sense, *incomplete*, because it considers only one (e.g., the most optimistic) scenario, which need not happen in real application-scenarios. If we drop the two assumptions and re-visit the problem, we have the following observations:

1. Although the object a has three choices at the intersection A, there is no way that it travelled along the route $(\overline{AC}, \overline{CD})$. Namely, even if it moved with the maximum speed along each segment, the earliest possible time for it to arrive at the sampled location p_2 would be $t'_2 = 8$, which is later then the actual sample-time.

2. Hence, we now have only two possible paths between the intersections: $\overline{AD}$ and $(\overline{AB}, \overline{BD})$.

   - If the object a travelled along $\overline{AD}$, it has an additional flexibility of moving a bit slower than the maximum speed. However, in order for it to make it at p_2 at $t_2 = 7$, it *must have been located somewhere within the segment $\overline{AF}$ at $t = 2$*. This, in turn, implies that *it is possible* for the object a to be within r from q at $t = 2$.
   - If the object a travelled along $(\overline{AB}, \overline{BD})$, the first observation is that it must have moved with the maximum speed throughout each segment (otherwise, it would not have reached p_2 at $t_2 = 7$). This, in turn, enables us to infer that at $t = 2$, the object a is located at the position G, which also falls inside the disk centered at q and with radius r.

By considering the uncertainty of the trajectory, we conclude that a has some *possibility* to qualify a result to Q_R. Besides, by modelling the uncertainty, we can also infer *how likely* it is for a to qualify. In this way, we can provide more insightful information to the users, which cannot be delivered by any traditional deterministic approach. Clearly, the situation becomes even challenging and complicated if one is interested in processing continuous range queries.

1.2 Challenges and Contributions

The above example demonstrates the two main sources of the uncertainty for objects moving on road networks:

1. Moving objects need not always follow the shortest travel-time/distance path between two locations. There is a variety of reasons, such as the driver preferring to drive near landmarks [25], a detour due to road works, etc.

2. In reality, one cannot expect that a moving object will maintain a constant speed along a given road segment. Such fluctuations will have to somehow be captured in the model and, consequently, the answer to a given range query.

Hence, in order to process a spatio-temporal range query for a collection of objects moving on a given road network, one needs to provide a model of the uncertainty and capture its impact on the query semantics. However, it brings challenges in many ways:

- *Uncertainty Model.* By eliminating the two assumptions of the deterministic approach, the location of a moving object becomes probabilistic. But as a complete uncertain trajectory model, we have to give the *probability distribution function* (PDF) of its location at all time instants. This is not an straightforward task, since the object's motion consists of uncountably many locations and time-instants.

- *Query Syntax and Semantics.* Since the locations of moving objects are probabilistic, whether an object should be returned for the query is not a binary decision any more. Intuitively, each object should be associated with a *qualification probability* to indicate how likely it is the answer of the

query. Only the ones having probabilities above a desired threshold should be returned. However, how to capture the qualification probability in the query syntax and semantics is still not clear.

- *Query Processing.* Real world applications often need to handle a large number of trajectories, making it prohibitive to individually process a given range query for each trajectory. Hence, effective indexing structure and efficient processing algorithms are critical for the applicability of a given solution.

In this paper, we represent the uncertain locations of objects moving along road networks as *time-dependent probability distribution functions (pdf)*, and then take a step towards developing techniques for efficient processing of *spatio-temporal range queries* incorporating the *qualification probability* of the objects belonging to the answer-set. In summary, following are our main contributions:

- We propose an intuitive model for *Uncertain Trajectory* representing the motion along a road network, and provide a unified *pdf* for the possible locations of a moving object at a given time-instant.
- We formulate both *Snap-shot* and *Continuous Probabilistic Range Query* whose semantics captures our uncertainty model.
- We design effective *indexing structure* as well as efficient *processing algorithms* for both snap-shot and continuous range queries.
- We conduct extensive experimental evaluation which demonstrates the benefits of the methodology proposed in this work.

The rest of this paper is organized as follows. In Section 2 we introduce the necessary background, and follow with proposing the uncertain trajectory model and formally defining the probabilistic range queries over the uncertain trajectories. In Section 3 we present the query processing methodology, beginning with indexing structure, and following with the pruning (verification) and refinement algorithms. Section 4 presents our experimental observations and Section 5 positions our work with respect to the related literature. Finally, in Section 6, we give some concluding remarks and outline directions for future work.

2. UNCERTAIN TRAJECTORY MODEL

In this section, we focus on the terminology necessary for the development of our main results. Firstly, we define the model of uncertain trajectories in road network used throughout this paper. Subsequently, we focus on how such trajectories can be constructed and represented in a MOD and, based on this model, we formulate the basic snap-shot probabilistic range query.

2.1 Road Networks and Uncertain Trajectories

Let $\mathbb{R}$ denote the set of the real number and $\mathbb{R}^2$ the 2D Euclidian space for which we assume a chosen reference coordinate system. Following the previous work on MOD-related modelling and querying of moving objects on road networks (e.g., [12, 14, 15, 21, 22, 26]), we adopt graphs for representing the road networks. Formally:

DEFINITION 1. *A road network is represented by a graph $G = (V, E)$ embedding in $\mathbb{R}^2$, where V and E are the set of vertices and edges, respectively. Each edge e is associated with an attribute vector $\langle l(e), s(e)\rangle$, corresponding to the length and maximum allowed speed of e.*

Note that Definition 1 implicitly represents the road network as an undirected graph, however, our approach can be straightforwardly extended to capture directed graphs. which has no impact on our algorithms. Another implicit information contained in Definition 1 is the value of $tc(e)$, which is, the minimum time-cost needed to travel through a given edge e, which can be calculated as $tc(e) = l(e)/s(e)$.

Ideally, a trajectory should correspond to a continuous map $f_T : \mathbb{R}^+ \rightarrow \mathbb{R}^2$, however, in reality, the GPS (and/or roadside sensor) based location samples can only be obtained at discrete time-instants. Hence, we have the notion of the *trajectory sample*, formally defined as:

DEFINITION 2. *A trajectory sample of a moving object a in road network G is a finite sequence of positions with timestamp: $TS_G(a) = \{(t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n)\}$, where, for $i = 1, \ldots, n$, p_i is the sample position in G at $t_i \in T$.*

Remark. The raw trajectory samples obtained from GPS devices are typically of the form of $(timestamp, longitude, latitude)$. How to align the $(longitude, latitude)$ pair onto the digital map of a given road network is an interesting research problem itself and outside the scope of this paper. We assume all the sampled locations have already been aligned on the road network by some map-matching algorithm [1, 2, 11, 31].

Typically, to complete the definition of the trajectory as a *map* from $\mathbb{R}^+$ to $\mathbb{R}^2$, we can assume that the moving objects always follow the shortest paths and travel at a constant speed between two consecutive trajectory samples. In this way, the position $p_a(t)$ of the object a on a road network G at the time-instant t ($t \in (t_i, t_{i+1})$) can be calculated as $p_a(t) = p_a(t_i) + \lambda d(p_a(t_i), p_a(t_{i+1}))$, where $\lambda = (t - t_i)/(t_{i+1} - t_i)$ and $d(p_a(t_i), p_a(t_{i+1}))$ denotes the *network-distance* between the sample locations at t_i and t_{i+1} along the shortest path. However, as we discussed in Section 1, these assumptions need not hold in real-world applications, hence the uncertainty inherently exists in the location of a moving object between trajectory samples. Unlike the *space-time prisms* [18, 23] where the 2D projection of the uncertain location correspond to an ellipse, uncertain trajectories models on road networks are different due to the constraints of movement [17].

Specifically, we now have two types of uncertainty.

2.1.1 Path Uncertainty

As illustrated by the Motivational Example in Section 1, when there are multiple paths connecting two positions p_i, p_{i+1} of moving object a, it is possible for a to move along a path, say P_j, which is different from the shortest path — for as long as P_j can be travelled through within the time period $[t_i, t_{i+1}]$. The path uncertainty can be more formally described by the notion of *valid possible path*.

DEFINITION 3. *Given two trajectory samples (t_i, p_i) and (t_{i+1}, p_{i+1}) of a moving object a on road network, the set of* possible paths *(PP_i) between t_i and t_{i+1} consists of all the paths along the routes (sequence of edges) that connect p_i and p_{i+1}, and whose minimum time costs are not greater than $t_{i+1} - t_i$, i.e.,*

$$PP_i(a) = \{P_j \in \mathit{Paths}(p_i, p_{i+1}) | tc(P_j) \leq t_{i+1} - t_i\},$$

where Paths denotes all the paths between p_i and p_{i+1}, $tc(P_j)$ is the sum of all the $tc(e)$ of $e \in P_j$.

Whenever clear from context, we also use $PP_i(a)$ to denote the random variable indicating the path taken by a between t_i and t_{i+1}. The probability distribution function of PP_i may be application dependent (e.g., uniform or weighted distribution) and does not affect our main ideas. For example, if the *pdf* of selecting a particular possible path is uniform, then:

$$Pr_{i,j}(a) = Pr[PP_i(a) = P_j] = \frac{1}{|PP_i|} \quad (1)$$

where $|PP_i|$ denotes the cardinality of the set of all the possible paths PP_i.

As another example, the probability of a particular path being taken by an object a can be inversely proportional to time-cost of that path, i.e.,

$$Pr_{i,j}(a) = Pr[PP_i(a) = P_j] = \frac{1/tc(P_j)}{\sum_{P_x \in PP_i(a)} 1/tc(P_x)} \quad (2)$$

Unless stated otherwise, in the rest of this paper we assume a uniform *pdf* for the possible location of an object at a given time-instant.

2.1.2 Location Uncertainty

For a particular $P_j \in PP_i$, we still cannot know the exact location of the moving object at a given time-instant $t \in (t_i, t_{i+1})$, because the velocity of the object along P_j need not be constant all the time. However, with the speed limit of each edge and, consequently, its time-cost available, we can restrict the location of the object within a segment of the path P_j.

DEFINITION 4. *Given a path $P_j \in PP_i(a)$, the* Possible Locations *of a given moving object a with respect to P_j at $t \in [t_i, t_{i+1}]$ is the set of all the positions $p \in P_j$ from which a can reach p_i (respectively, p_{i+1}) within time period $t - t_i$ (respectively, $t_{i+1} - t$) i.e.,*

$$PL_{i,j}(t) = \left\{ p \in P_j \;\middle|\; \begin{array}{l} tc_{P_j}(p_i, p) \le t - t_i \\ tc_{P_j}(p, p_{i+1}) \le t_{i+1} - t \end{array} \right\} \quad (3)$$

Clearly, $PL_{i,j}(t)$ is a continuous subset of P_j. Once again, the *pdf* of the object's whereabouts along $PL_{i,j}(t)$ is likely to be application-dependent, e.g., it can be uniform across the entire $PL_{i,j}(t)$, or it can vary along different (portions of the) edges comprising $PL_{i,j}(t)$. Let $\overline{PL_{i,j}}(t)$ denote the network-length of $PL_{i,j}(t)$. In the case of a uniform *pdf*, the probability that the object a falls inside some portion $[p_A, p_B] \subset PL_{i,j}(t)$, whose network-distance is $d(p_A, p_B)$, can be calculated as:

$$Pr[p_a(t) \in [p_A, p_B]] = Pr_{i,j}(a) \cdot \frac{d(p_A, p_B)}{\overline{PL_{i,j}(t)}} \quad (4)$$

Intuitively, the above formula states that when calculating the probability of an object a being somewhere along the segment $\overline{p_A, p_B}$ at a given time-instant t, one needs jointly consider the probability of P_j being selected from among the possible paths.

The concepts introduced so far can be illustrated by the following:

EXAMPLE 1. *Consider the trajectory samples of moving object a in road network as shown in Figure 2. For illustration purpose, we use a unit grid (the side of each cell is 1) as measurement and assume the maximum speeds of all edges are 1, hence, the travel-time cost is also 1 for each edge. Since the minimum time costs of both P_1 and P_2 are the same, the set PP_i consists of both paths with equal probability, $PP_i(a) = \{(P_1, 50\%), (P_2, 50\%)\}$, regardless of whether the pdf is uniform or weighted by the respective tc's. Suppose that we want to determine the possible locations of a at $t = 4$. If a moves along P_1, it can reach as far as p_1 (with maximum speed). On the other hand, it has to arrive at least p_2 when $t = 4$ since otherwise it cannot reach the next sample position when $t = 7$. Thus, the segment $\overline{p_1 p_2}$ is the probabilistic location $PL_{i,1}(4)$ of a when P_1 is chosen. Similarly, we can infer the probabilistic location when P_2 is chosen (shown as the blue segment along P_2). In accordance with the formula 4 above, the probability that the object a will be somewhere between the point A, the midpoint of $\overline{p_1, p_2}$, and p_1 in Figure 2 (darker, thicker region) can be calculated as 0.5 · 0.5 = 0.25.*

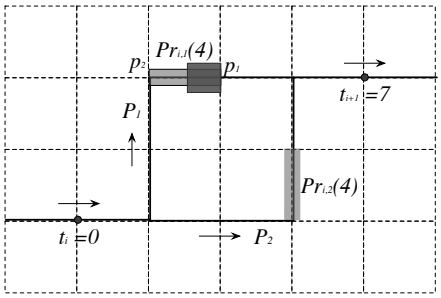


Figure 2: Illustration of path and location uncertainty

We are now in position to formally define *uncertain trajectories on road networks*.

DEFINITION 5. *Given a trajectory sample $TS(a) = \{(t_1, p_1), (t_2, p_2), \ldots, (t_n, p_n)\}$ of moving object a on road network G, the uncertainty trajectory $UT(a) : t \to PL_a$ is a mapping from time instant $t \in T$ to the probabilistic location on G, i.e., $\forall i = 1, 2, \ldots, n$,*

$$PL_a(t) = \begin{cases} p_i, & t = t_i \\ PL_i(t), & t \in (t_i, t_i + 1) \end{cases} \quad (5)$$

where $PL_i(t)$ is the union of all probabilistic locations on each possible path, i.e., $PL_i(t) = \bigcup_{P_j \in PP_i(a)} PL_{i,j}(t)$.

2.2 Uncertain Trajectory Construction

While Definition 5 provides a declarative way of defining uncertain trajectory based on the discrete set of a given trajectory sample, it provides no procedural means on constructing a structure that can be used to store uncertain trajectories in the MOD. The two basic problems are:

1. Given the road network and the trajectory sample, how can one obtain the set of possible paths.
2. Given that the points along the road networks, as well as the time-instants in-between consecutive samples are uncountably infinite, how can one provide a finite/discrete representation for the probabilistic locations at all time instants.

2.2.1 Generating Possible Paths

Given any two consecutive trajectory samples (t_i, p_i) and (t_{i+1}, p_{i+1}) of moving object a, in order to generate the set PP_i(a), we proceed

as follows. Assume that we have a "virtual object" a' which travels from p_i at maximum speed for a period of $t_{i+1} - t_i$. Whenever a vertex in the graph is encountered which has > 2 incident edges, a' it will try every adjacent edge except the one it comes from. Finally, all the routes passed by a' that can successfully reach p_{i+1} in time will form the set of PP_i.

The idea, in a sense, amounts to finding all possible paths between p_i and p_{i+1}, which are below certain cost $(t_{i+1} - t_i)$. To implement this, we use a priority queue (heap) H which maintains the explored paths along with the minimum time costs for passing through them. In each successive round, the algorithm will retrieve a path P from the top of H and expand P by using its adjacent edges that have not been visited yet. There are two distinct cases to be handled:

- If p_{i+1} is on e, only the partial edge needs to be appended to P. Then we output P as a possible path if its time cost is not greater than $(t_{i+1} - t_i)$.
- Otherwise, we just append e to P and insert it back to H.

This process is repeated until the time cost of the top element in H is greater than $(t_{i+1} - t_i)$, or the heap becomes empty. Note that for the convenience of illustration, we assume that an moving object travels via a simple path in between consecutive time instants. However, the proposed method can also handle the cyclic path with slightly modification.

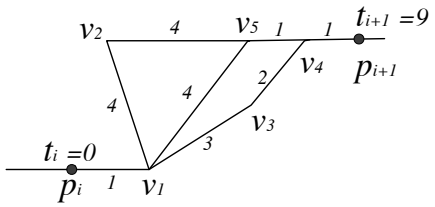


Figure 3: Generating possible path

EXAMPLE 2. *To explain this process, consider a fraction of road network shown in Figure 3, where the number beside each edge is its minimal time cost. During the search for the PPs between p_i and p_{i+1}, the dynamic content of the heap H is shown in the following table.*

Iteration	Content of H: $\langle path, cost \rangle$
1	$\langle p_i \rightarrow v_1, 1 \rangle$
2	$\langle p_i \rightarrow v_1 \rightarrow v_3, 4 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_2, 5 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_5, 5 \rangle$
3	$\langle p_i \rightarrow v_1 \rightarrow v_2, 5 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_5, 5 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_3 \rightarrow v_4, 6 \rangle$
4	$\langle p_i \rightarrow v_1 \rightarrow v_5 \rightarrow v_4, 6 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_3 \rightarrow v_4, 6 \rangle$
5	$\langle p_i \rightarrow v_1 \rightarrow v_5 \rightarrow v_4 \rightarrow p_{i+1}, 7 \rangle$ $\langle p_i \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow p_{i+1}, 7 \rangle$

2.2.2 Probabilistic Location Function

After all the elements of PP_i have been obtained, and for every i in the trajectory sample, we still need to quantify the probabilistic locations of a. In other words, a *probabilistic location function (PLF)* is required for every time instant in-between samples. As we mentioned, there are uncountably many of them and, to explain our methodology, consider a particular path $P_j \in PP_i$. Suppose P_j sequentially connects the following $n+1$ points: $v_{i,j}^0 \rightarrow v_{i,j}^1 \rightarrow \ldots \rightarrow v_{i,j}^n$, where $v_{i,j}^0 = p_i, v_{i,j}^n = p_{i+1}$; and all the other $v_{i,j}^k$ $(k \notin \{0, n\})$ are vertices of the graph from the underlying road network. Extending Definition 5, observe that at any $t \in [t_i, t_{i+1}]$, the *PLF* of a can be represented by an upper and lower bound function, namely $PLF_{i,j}^+(t)$ and $PLF_{i,j}^-(t)$, indicating the furthest and nearest position that a can reach from p_i along P_j.

$$PLF_{i,j}(t): \begin{cases} PLF_{i,j}^+(t) = \{p \in P_j | tc_{P_j}(p, p_i) = t - t_i\} \\ PLF_{i,j}^-(t) = \{p \in P_j | tc_{P_j}(p, p_{i+1}) = t_{i+1} - t\} \end{cases} \quad (6)$$

Due to the continuity of the time instants we cannot store all such *PLF*s. However, we observe it suffices to store just a finite (discrete) collection of *vertex-critical time instants*. To explain this, we use $t_{ea}(v_{i,j}^k)$ and $t_{ld}(v_{i,j}^k)$ to denote the *earliest arrival time* and *latest departure time* for a given vertex $v_{i,j}^k$ $(k \in \{0, 1, \ldots, n\})$ along P_j. Trivially, $t_{ea}(v_{i,j}^0) = t_i$ and $t_{ld}(v_{i,j}^n) = t_{i+1}$. Then the *earliest arrival time* and *latest departure time* of other vertices can be inferred recursively as follows.

$$t_{ea}(v_{i,j}^k) = t_{ea}(v_{i,j}^{k-1}) + tc(v_{i,j}^{k-1}, v_{i,j}^k) \quad (7)$$

$$t_{ld}(v_{i,j}^k) = t_{ld}(v_{i,j}^{k+1}) - tc(v_{i,j}^k, v_{i,j}^{k+1}) \quad (8)$$

Then, for a given P_j, both $PLF_{i,j}^+$ and $PLF_{i,j}^-$ are piece-wise linear functions which can be obtained by sequentially connecting the pairs $(t_{ea}(v_{i,j}^k), v_{i,j}^k)$ and $(t_{ld}(v_{i,j}^k), v_{i,j}^k)$ for $k = 0, 1, \ldots, n$ respectively. The concepts of $t_{ea}(v)$ and $t_{ld}(v)$ are illustrated in the following:

EXAMPLE 3. *Figure 4 shows the PLF for the two Possible Paths (derived at the last iteration) in the example from Figure 3. For any time instant t, the bold line segments bounded by PLF^- and PLF^+ represent the probabilistic locations of the moving object. Note that in the locations p_i and p_{i+1}, the length of such segments is 0, since those locations correspond to the actual samples.*

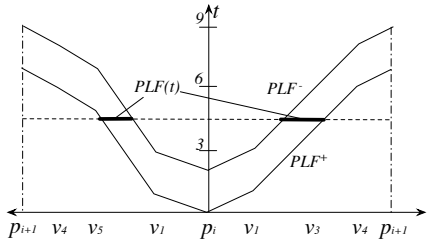


Figure 4: The probabilistic location function

2.2.3 Snap-shot Probabilistic Range Query

We now formally define the basic form of the *probabilistic range queries* for uncertain trajectories on road networks, pertaining to a particular time-instant.

DEFINITION 6. *Let D denote a MOD of uncertain trajectories on a road network G, $q \in G$ denote a query-point, $r \in \mathbb{R}$ denote a range, t_q denote a (query) time-instant, and $\alpha \in [0, 1]$ denote a probability threshold. The answer to the* snap-shot probabilistic range query SPR(q, t_q, r, α, D) *consists of all the objects $a \in D$ whose* qualification probability (QP) *of being within network distance r from q at t_q, is not less than α.*

Let $pdf_a(x)$ denote the probability density function that the moving object a is at the location x at t_q. For the above definition, the *qualification probability (QP)* of a being within network distance r with respect to q at a given time instant $t_q \in (t_i, t_{i+1})$, can be calculated as:

$$QP^r_{a,q}(t_q) = \sum_{P_j \in PP_i} Pr[PP_i = P_j] \cdot \int_{x \in \{p' \in P_j | d(p',q) \le r\}} pdf_a(x)dx \quad (9)$$

We conclude this section with a table providing a summary of the symbols introduced so far.

Notation	Definition
G	graph representing the road network
v	vertex of G
e	edge of G
p	a position on G
P	a path of G
$TS(a)$	trajectory sample of moving object a
$d(p,q)$	network distance between p and q
$d_P(p,q)$	distance between p and q along path P
$tc(p,q)$	minimum time cost from p to q
$tc_P(p,q)$	minimum time cost from p to q along path P
PP_i	possible path between the i-th and $i+1$-th samples
$PL_i(t)$	the probabilistic location at $t \in (t_i, t_{i+1})$
$PLF(t)$	the probabilistic location function
$QP^r_{a,q}$	Qualification Probability of a within distance r from q

Table 1: Summary of notations

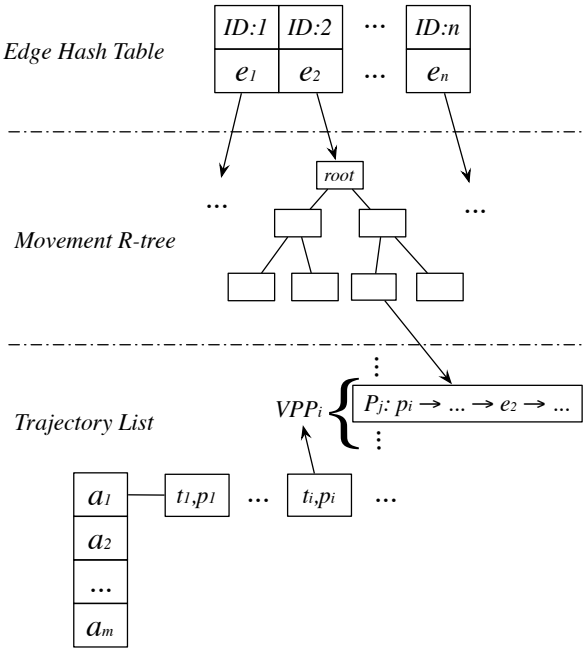


Figure 5: Uncertain Trajectory Hierarchy

So the indexing structure should be augmented to capture this uncertainty.

3. PROCESSING PROBABILISTIC RANGE QUERIES

In this section, we present our results for processing probabilistic range queries for uncertain trajectories on road networks. Our approach follows the traditional query processing steps of *filtering + refinement*. In the sequel, we first propose a novel indexing structure, *Uncertain Trajectory Hierarchy*, to organize the trajectories effectively and to use it in the filtering stage. Subsequently, we focus on the details of the actual processing techniques for snapshot range queries, followed by continuous range queries.

3.1 Indexing Structure

To index the uncertain trajectories, we borrow some ideas from FNR-Tree [9] which is proposed to organize moving objects on road networks. An FNR-Tree consists of a top level 2D R-Tree whose leaf entries contain pointers to 1D R-Trees. The 2D R-Tree is used to index the edges of the network, for each of which there is an 1D R-Tree indexing the time interval of the objects traversing it. However, we will not adopt this structure directly because:

- The purpose for the FNR-Tree using a 2D R-Tree in top level is to find the edges intersected with the query window in Euclidean space efficiently. But in our problem, the edge where the query point falls can be determined by its unique identifier, in which means, using R-tree in top level is inefficient in our case.
- The FNR-Tree is developed to index the trajectories of deterministic moving objects, for which we know about their precise locations at all time instants. But in this paper we are handling uncertain trajectories, which means the exact time interval of the objects traversing each edge is not available.

To this end, we propose a novel indexing structure, *Uncertain Trajectory Hierarchy* (UTH), to index the road network, object movement and trajectories in a hierarchical style. It consists of three major components and its structure is demonstrated in Figure 5.

We now explain the main components of the *UTH*.

Edge Hash Table. Since in our problem setting, all the positions are given in the form of edge ID and offset, there is no need to index the edges by R-tree. To retrieve an edge by its ID effectively, we simply adopt hash table that can fit into the main memory even for a large road network (e.g., tens of thousands of edges).

Movement R-tree. To quickly determine if an edge contains the candidate objects of interest, we maintain an 1D R-tree (mR) to organize the time periods within which some objects are moving on it. For an edge $e = (v_i, v_j)$, its mR can be constructed as follows. If e belongs to a possible path of some object a, an entry will be inserted into mR along with $[t_{ea}(v_i), t_{ld}(v_j)]$, indicating the *maximum time interval* (MTI) for a being on e when following this path. Please note that each entry corresponds to the movement of certain possible path. So if e is contained by multiple possible paths of the same object, more than one entries will be inserted. Besides, a pointer to the path is also recorded in the entry to speed up the search. Based on the size of road network and number of trajectories, this component may be stored in memory or disk or both.

Trajectory List. This component stores the actual trajectory data. For each moving object a, its trajectory samples are sorted based on timestamps. Each trajectory sample $TS_i(a)$ points to the set PP_i. For each path $P_j \in PP_i$, t_{ea} and t_{ld} of all vertices it traversed are recorded so that $PLF_{i,j}(t)$ can be obtained efficiently. This part is usually stored on disk due to the high space requirement, and retrieved on demand (usually in the refinement phase of query processing).

To build the UTH, we first initialize an empty movement R-tree for each edge. Then we compute the possible paths for all the moving objects. During this process, whenever an edge e is encountered and its MTI becomes available, an entry is inserted into the movement R-tree of edge e.

3.2 Processing Snap-shot Range Queries

The first observation for processing a given range query is that we only need to consider the moving objects whose possible paths is within the query range at some time between the query time interval. In this light, the snap-shot range queries can be processed in the filtering – refinement style. Specifically, it can be conducted in the following three steps.

Range Network Expansion. Since the range r is explicitly specified in the syntax of the query, as a first step we compute all the affected edges, which is, the ones that contain some parts with $\leq r$ network distance from q. This can be done by finding the *expansion tree* [21] of q, denoted by $ET(q, r)$, which is a tree rooted at q and containing all the positions along the edges of G that are within network distance of r from q.

Filtering. Only those moving objects that travel inside $ET(q, r)$ at the query time instant may belong to the final result. Based on this, we can prune or filter most disqualifying objects with the help of the UTH structure. Particularly, for each edge $e \in ET(q, r)$, we query the $mR(e)$ to retrieve all the entries with $t_q \in MTI$. Then all the paths pointed by these entries form the set of the *candidate paths (CP)*. Typically, compared to the original data set, the size of CP is significantly smaller, which means all the candidate paths can be stored within the main memory for the refinement stage.

Refinement. This step calculates the exact qualification probability for each candidate and returns all the objects that satisfy the probability threshold. First, we group the paths in CP by the objects they belong to. Apparently, the paths in the same group belong to the same PP_i of some object a. Before calculating the exact $QP^r_a(t_q)$, we evaluate the sum of the individual probabilities of its PP's in CP first. If $\sum_{P_j \in PP_i \cap CP} Pr[PP_i = P_j] < \alpha$, a can be rejected immediately since it is impossible for its QP to be greater than α.

Computing the QP by Equation (9) is likely to involve expensive numerical integration. To avoid that, we use the *affecting fraction* of a given P_j, defined as $AF_{P_j} = P_j \cap ET(q, r)$. An illustration is provided in Figure 6, which we plot AF_{P_j} together with the $PL_{i,j}(t)$. The following Lemma provides means for calculating the QP of a given object a via simpler algebraic operations.

LEMMA 1. *The qualification probability of an object a conditioned on P_j can be calculated as follows:*

$$QP^r_{a,q}(t) = \sum_{P_j \in PP_i \cap CP} [QP^r_{a,q}(t)|P_j] \cdot Pr[PP_i = P_j] \quad (10)$$

$$QP^r_{a,q}(t)|P_j = \frac{|PL_{i,j}(t) \cap AF_{P_j}|}{|PL_{i,j}(t)|} \quad (11)$$

3.3 Continuous Probabilistic Range Queries

We now focus on the continuous variant of the probabilistic range queries. Depending on the dimension to be extended (time or location), we distinguish between *temporal-continuous* and *spatio-continuous* counterparts. In theory, a continuous query can always be answered by repeatedly issuing a series of static queries. However, this approach can either be very inefficient (when the granularity is too fine), or may distort the (changes of the) answer as the continuous dimension evolves. In the sequel, we propose efficient algorithms for both continuous queries, which use as few static queries as possible and reduce the computation overhead in refinement step by utilizing the relationship between the QP and query time/location.

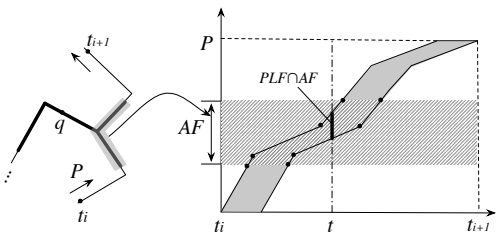


Figure 6: Evaluation of qualification probability

3.3.1 Temporal-Continuous Range Query

First, the *Temporal-Continuous Range Query* is defined as follows:

DEFINITION 7. *Let D denote a MOD of uncertain trajectories on a road network G, $q \in G$ denote a query-point, $r \in \mathbb{R}$ denote a range. Also, let $T_q = [t_{qs}, t_{qe}]$ denote a (query) time-interval, and $\alpha \in [0, 1]$ denote a probability threshold. The answer to the Temporal-Continuous Probabilistic Range Query TCPR(q, T_q, r, α, D) consists of all the tuples of the form (a, T_a), where $a \in D$ and $T_a \subseteq T_q$ is an interval (called valid period) during which a is in the answer-set of SPR(q, t_q, r, α, D) for every $t_q \in T_a$.*

To answer the TCPR query, we can still adopt the *expansion-filtering-refinement* framework. Since the first step (expansion) is exactly the same as explained before, we now elaborate the last two steps of the processing.

Filtering. In this step, we search for the candidate objects that possibly travel in the expansion tree during the query time-interval T_q. Hence, we use $[t_{qs}, t_{qe}]$ as a window for querying the mR for each edge $e \in ET(q, r)$, thus retrieving all the entries with $MTI \bigcap [t_{qs}, t_{qe}] \neq \emptyset$. All the paths pointed by these entries are added into a candidate path set (CP) and passed to the refinement step.

Refinement. A naive refinement method is to first divide the query time interval into a finite set of time instants (granularity depending on precision requirement), and calculate the QP for each candidate object at each $t \in [t_{qs}, t_{qe}]$. Assuming some "light" continuity requirements, this method will work, however, it may be very inefficient for practical purposes because it may require a large number of QP calculation.

To improve the refinement performance, we propose a plane-sweep approach. It is based on the observation that, instead of sweeping through all the (uncountably many) time-instants within query time interval, it suffices to process a few *critical time instants* by taking advantage of the monotonicity of the QP function. For presentation convenience, and without loss of generality, in the sequel we focus on presenting our algorithm on a given path $P \in CP$ (cf. Figure 6), for which we assume its entire time period to fall inside $[t_{qs}, t_{qe}]$.

Suppose the qualification fraction of P intersects with the PLF at a set of time instants, denoted by T_1. Let the set T_2 consist of all the t_{ea} and t_{ld} of each vertex along P. Then, the set of *critical time instants* (CTI) of the given TCPR is defined to be the union of T_1 and a subset of T_2, defined as:

$$CTI = T_1 \bigcup \{t \in T_2 | \min T_1 \leq t \leq \max T_1\}$$

By sorting CTI in ascending order, $\{t_{i,1} \leq t_{i,2} \leq \ldots \leq t_{i,k}\}$, the QP function for any $t \in [t_{i,j}, t_{i,j+1}]$ can be expressed in the form of $(a \cdot t + b)/(c \cdot t + d)$ where a, b, c, d are constants. This, in turn, ensures that the QP function is of a fixed monotonicity (either increasing or decreasing) during consecutive time-instants of CTI. In the light of this observation, an envelop function that consists of upper and lower bounds of QP can be derived as:

$$QP^{+}(t) = \max\{QP(t_{i,j}), QP(t_{i,j+1})\}, t \in [t_{i,j}, t_{i,j+1}]$$
$$QP^{-}(t) = \min\{QP(t_{i,j}), QP(t_{i,j+1})\}, t \in [t_{i,j}, t_{i,j+1}]$$

The plane-sweep algorithm will process the member of CTI sequentially. Whenever the plane "encounters" the next $t_{i,j}$, the actual QP for that time-instant is calculated. Together with $QP(t_{i,j-1})$, the envelop function during $[t_{i,j-1}, t_{i,j}]$ can be obtained. Then the envelop of QP is compared with α. If $QP^{+} < \alpha$ (or $QP^{-} > \alpha$), $[t_{i,j-1}, t_{i,j}]$ can be rejected (or accepted) as a qualifying time interval without knowing the exact QP. Otherwise, there must be a t in this period that $QP(t) = \alpha$, which can be obtained t using numerical methods.

The envelope function is exemplified in Figure 7. Only when α is between the lower and upper bounds of QP should we calculate the exact probabilities by numerical analysis, which considerably reduces the refinement cost of *TCPR*.

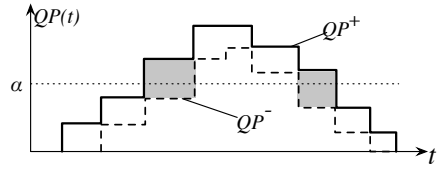


Figure 7: Envelop of qualification probability

Complexity: Finally, we give a complexity analysis on the refinement phase for the TCPR query processing, which is equivalent to estimating the average number of *critical time instants* that our plane-sweep algorithm will encounter. Let m and $|PP|_{avg}$ denote the average time span and number of possible paths between two samples, tc_{avg} denote the average time cost of an edge. Then the number of samples covered by query time interval T_q is $|T_q|/m$. Together with the average number of edges in each possible path, i.e., m/tc_{avg}, we have the overall refinement complexity for each candidate is $O(\frac{|T_q|}{m} \cdot |PP|_{avg} \cdot \frac{m}{tc_{avg}}) = O(|T_q| \cdot |PP|_{avg}/tc_{avg})$.

3.3.2 Spatio-Continuous Range Query

The second kind of continuity for the answer(s) to a range query is formalized by:

DEFINITION 8. *Let D denote a MOD of uncertain trajectories on a road network G, $r \in \mathbb{R}$ denote a range and t_q denote a query time-instant. Also, let $P_q \in G$ denote a query path, and $\alpha \in [0, 1]$ denote a probability threshold. The answer to a* spatio-continuous probabilistic range query SCPR(P_q, t_q, r, α, D) *consists of all the tuples (a, I_a), where $I_a \subseteq P_q$ is called a* valid path-interval *over which $(\forall q \in I_a) QP^{r}_{a,q}(t_q) \geq \alpha$.*

A straightforward approach to process *SCPR* query is to divide the entire query path into a finite set of discrete locations that satisfy some application-dependent precision requirement. Then, a static *PR* query is issued for each location on the query path to find the qualifying objects. Finally the locations with the same qualifying objects are merged to form the valid interval. However, this is not an efficient approach since it usually requires a large number of evaluations of static queries, increasing with the length of P_q.

As with any scalable algorithm, we want to avoid re-issuing the static query as much as possible. In the sequel, we explain our algorithm for a single edge $e \in P_q$. Without loss of generality, we also assume $r \geq l(e)/2$ since otherwise we can always (recursively) split e until it satisfies this condition.

We now proceed with explaining in detail all the steps of processing *SCPR* queries.

Network Expansion. Unlike the static *PR* query, we need to compute the expansion trees for all the locations in e, i.e., $ET(e, r) = \{ET(q, r) | q \in e\}$. However, independently computing all the expansion trees for each location is prohibitive. To avoid this, we observe it is equivalent to an expansion tree from c_e with range $r + l(e)/2$, where c_e is the center of e – a property formalized with the following:

LEMMA 2. *The two expansion trees are equivalent, i.e., $ET(e, r) \Leftrightarrow ET(c_e, r + l(e)/2)$*

PROOF. It suffices to prove that $\forall p \in ET(e, r), p \in ET(c_e, r + l(e)/2)$ and vice versa.

- $\forall p \in ET(e, r)$, there must exists a point $q \in e$ such that $p \in ET(q, r)$. Since c_e is the center of e, $d(c_e, p) \leq l(e)/2$. So $d(c_e, p) \leq r + l(e)/2$ which means p belongs to $ET(c_e, r + l(e)/2)$.
- $\forall p \in ET(c_e, r + l(e)/2)$, if $p \in e$, then $d(c_e, p) \leq l(e)/2 \leq r$; otherwise there must be one end vertex v_e of e such that $d(p, v_e) \leq r$. In either case, $p \in ET(e, r)$.

□

Filtering. This step is exactly the same as the static *PR* query processing, except the $ET(q, r)$ is replaced by $ET(e, r)$. Apparently, only the objects returned in this step have a chance to satisfy the query-threshold for some $q \in e$.

Refinement. This phase finally determines the respective valid intervals for all the candidate objects. Consequently, reducing any computational overhead of this step is of interest for improving the overall performance of the *SCPR* query processing.

To illustrate the intuition behind our refinement procedure, consider a candidate object a whose probabilistic locations at t_q on the road-network graph are represented as bold lines in Figure 8a. In addition, the expansion trees from the end-vertices of e, v_1 and v_2, shown as grey region and dashed region, respectively. We denote the part of $ET(v_1, r)$ with and without edge e as $ET_e(v_1, r)$ and $ET_{\not e}(v_1, r)$ – and similarly, we use $ET_e(v_2, r)$ and $ET_{\not e}(v_1, r)$. Imagine a query point q moving from v_1 towards v_2, during which the length of $ET_{\not e}(v_1, r)$ is shrinking and $ET_{\not e}(v_2, r)$ is expanded (in synchronized manner). By defining

$$(i) IP_e = ET_{\not e}(v_1, r) - ET_e(v_2, r)$$
$$(ii) VP_e = ET_{\not e}(v_2, r) - ET_e(v_1, r)$$

we get two path-collections on which the locations of a will gradually become: (i) invalid; and (ii) valid, as q moves from v_1 towards v_2. As for the rest of $ET(e, r) - (IP_e \cup VP_e)$, we observe that it is always covered by the expansion tree of q. Consequently, the locations of a in this part are always valid (e.g., $PL_{i,2}$).

What we need now is to somehow capture the variations of $QP_{a,q}(t_q)$ when q moves from v_1 towards v_2. Suppose that the intersection of $PL_{i,j}$ and IP_e (or VP_e) has been changed (either shrank or expanded) by some distance x. From Equation (10), we obtain the

change of the *QP* to be $\Delta(x) = x \cdot Pr_{i,j}(a)/PL_{i,j}(t_q)$. We note that all the intersections can be represented via such *variation functions*, where x-axis represents the offset of the q's motion, while y-axis represents the variation of the corresponding *QP*.

For a given intersection-segment I that is located along a path $P : p_1 \rightarrow p_2 \in IP_e(VP_e)$, if the both ends of I have x_1 and x_2 distance to p_1, the corresponding variation function of I is a line segment with x-coordinate ranging from x_1 to x_2 and slope of $-(+)Pr_{i,j}(a)/PL_{i,j}(t_q)$. As an illustration, the corresponding variation function of I_1 and I_2 from Figure 8a are shown in Figure 8b. As illustrated, when q moves from v_1 towards $(v_1, x_1/l(e), v_2)$, the QP of a is decreasing with the rate of $Pr_{i,j}(a)/PL_{i,j}(t_q)$. When q moves from $(v_1, x_2/l(e), v_2)$ towards v_2, the QP of a is increasing with the same rate. Note that when q moves between $(v_1, x_1/l(e), v_2)$ and $(v_1, x_2/l(e), v_2)$, the two variation functions overlap. In this case, their effects can be combined, resulting a new function with the slope equal to the sum of the ones for the variations functions of I_1 and I_2. In our example, the slope is zero which means the QP of a does not change in the overlapping segment.

Given the variation functions, we proceed by sorting the *start* and *end* x-coordinates of all the line segments and process them sequentially. For each query location q_{x_i}, we combine all the variation functions overlapping at this point. By using current QP and the overall variation slope, the QP with respect to $q \in [q_{x_i}, q_{x_{i+1}}]$ can be derived straightforwardly, which enables us to determine the valid interval of a for $q \in e$. Finally, to obtain the entire I_a, we just repeat the above process for each $e \in P_q$.

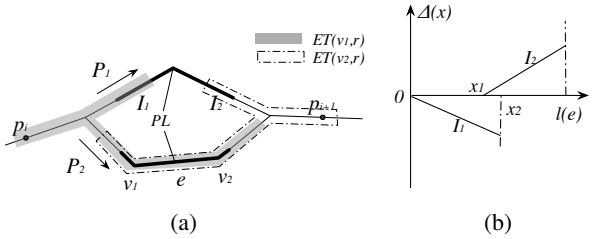


Figure 8: SCPR refinement

Complexity: Again we estimate the average number of static queries issued in the refinement step of SCPR query processing. Let l_{avg} denote the average length of an edge. Then the average number of edges covered by the query path P_q is $|P_q|/l_{avg}$. For each of them, the number of variation functions, as well as the QPs that we need to be evaluated is at most equal to the number of possible paths in PP_i. Therefore, the time complexity of the refinement stage for each candidate is $O(|P_q| \cdot |PP|_{avg}/l_{avg})$.

4. EXPERIMENTAL OBSERVATIONS

We report the results of the empirical study of the benefits of our proposed models and methodologies. All experiments are implemented in Java on a Pentium IV 2.4 GHz, 1GB memory and Windows XP platform.

4.1 Experiment Setup

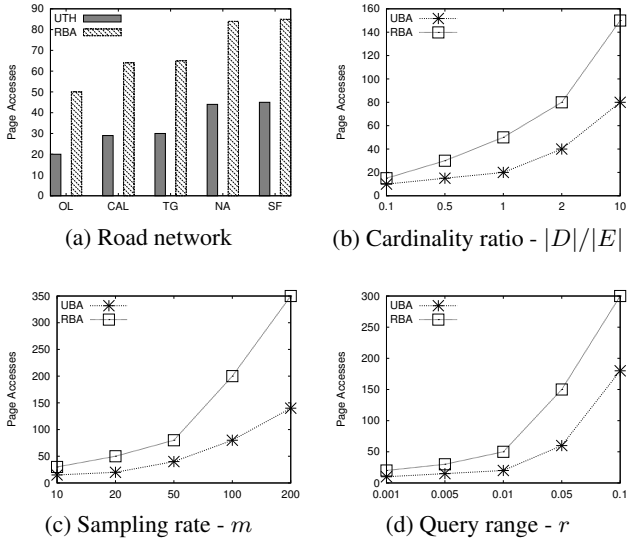
We use the following five real road-network data sets for our experiments: City of Oldenburg Road Network (OL) (6,104 nodes and 7,034 edges), City of San Joaquin County Road Network (TG) (18,262 nodes and 23,873 edges), California Road Network (CAL) (21,047 nodes and 21,692 edges), San Francisco Road Network (SF) (174,955 nodes and 223,000 edges), and North America Road Network (NA) (175,812 nodes and 179,178 edges). The length of each edge is the Euclidean distance between its end nodes, and the speed limit is set to be $s(e) = l(e)/5$ (proportional with the edge length).

The moving objects trajectories are generated as follows. For each trajectory, two vertices of the road network are selected randomly as its source and destination. Then we compute the top K shortest paths between them, one of which is selected as the "ground truth". The motivation behind this is that moving objects need not necessarily follow the shortest path strictly. After that, we simulate a moving object to travel along this path at the speed varying within $[s/2, s]$, and we allow the speed to change only when the object switches edges. Its trajectory sample is obtained by recording the location every m time-instants. As an illustration, Table 2 shows an instance of the parameter-space used in the experiments.

Parameter	Default value
Road network	OL
Cardinality ratio	1
Sampling rate m	50
Query range r	0.01 of data universe side length
Probability threshold α	0.5
Query time span	500
Query path length	200

Table 2: Parameter settings

4.2 Snap-shot Range Queries



(a) Road network (b) Cardinality ratio - $|D|/|E|$

(c) Sampling rate - m (d) Query range - r

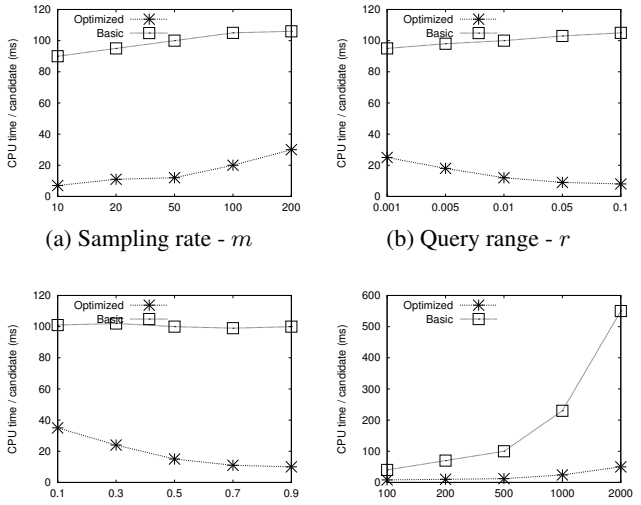
Figure 9: Performance of range queries

In this set of experiments, we compare the I/O performance in the filtering phase of the UTH based approach (UBA) against an R-tree based approach (RBA). RBA adopts a single 3D R-tree (*X-Y-time*) to index the sample locations of the trajectories. For a range query with time instant t_q and range r, we use $[t_q - m, t_q + m]$ as the time window and $r + m \cdot S_{max}$ as the working radius to search the R-tree for candidates, where $S_{max} = \max\{s(e)|e \in E_G\}$ is the global maximum speed of the road network. It is guaranteed that RBA will return all the candidates without introducing false negatives, since any object having no sample position inside this

range cannot be within network distance r to q at t_q even it travels at the maximum possible speed (note here we utilize the property that the Euclidean distance lower bounds the network distance).

From Figure 9a we can see that the UBA constantly outperforms the RBA in all road networks. Observe that both approaches increase the number of I/O accesses as the networks grow larger. For RBA, this is simply because the R-tree grows bigger as there are more trajectories. For UBA, this is due to the fact that, with the increasing number of edges in the road network, more movement R-trees cannot reside in the memory. Figure 9b plots the I/O performance of the two methods as a function of $|D|/|E|$. As the trajectory density increases, more objects may become candidates, which explains the fact that more page accesses are issued. On the other hand, the gap of two algorithms increases with the ratio since the R-tree based approach retrieves more false hits (i.e., objects in the Euclidean, but not in the network range), which results in more R-tree node accesses. Figure 9c shows the respective costs of the approaches as the sampling rate decreases. Since, in this case, the movement uncertainty increases, more objects will be included as candidates for a fixed query range, resulting in more page accesses for UBA. For the RBA, increasing m means a wider time window and larger search range, both of which will cause I/O cost boost quickly. Lastly, we tested the performance of both approaches for different query-ranges. As shown in Figure 9d, I/O cost increases as the query range expands, since UBA needs to search more edges as well as their associated R-trees, whereas RBA needs to access more R-tree nodes too, and at a much higher rate than UBS.

4.3 Continuous Range Queries



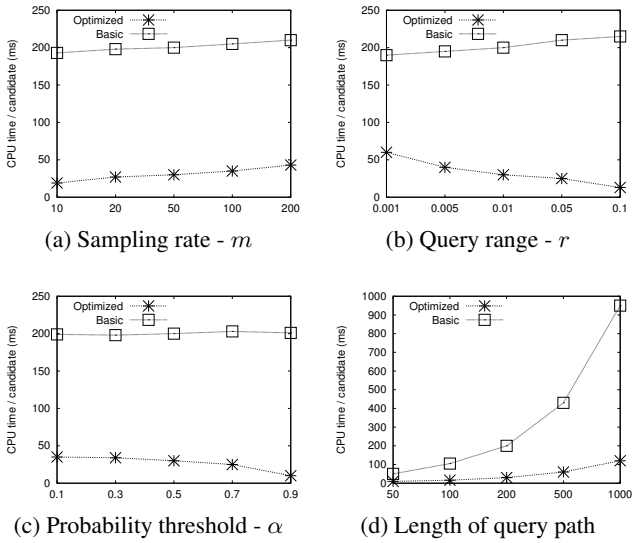
(a) Sampling rate - m (b) Query range - r

(c) Probability threshold - α (d) Length of query time interval

Figure 10: Performance of TCPR queries

In this subsection, we compare the performance of our proposed refinement strategy, termed *Optimized*, with the *Basic* method which breaks the query interval (time or path) into a set of snapshot-instances and evaluates the qualification probability individually. To eliminate the effects of candidate size, we measure the average processing time for each candidate. As the performance improvements of *TCPR* and *SCPR* queries exhibit similar behaviors, we combine the discussion of the respective results in the sequel.

As illustrated in Figure 10a and Figure 11a, *Optimized* consistently outperforms *Basic* at all sampling rates. Observe that the



(a) Sampling rate - m (b) Query range - r

(c) Probability threshold - α (d) Length of query path

Figure 11: Performance of SCPR queries

running time of each method increases, although not drastically, with m. For the *Basic* method, this can be caused by the more possible paths, which makes the probability evaluation more expensive. For the *Optimized* scheme, this can be explained by the fact that the probabilistic location function becomes more complicated, resulting in more re-evaluations at the respective change points.

Figure 10b and Figure 11b show the comparison of the processing time of both strategies as functions of query range. While *Basic* method is almost immune to this parameter, the performance of *Optimized* method improves for larger ranges. This is because more candidates are having their uncertain locations (almost) totally covered by the query range, and their qualification probability can be evaluated quickly due to the fewer change points.

From Figure 10c and Figure 11c, we observe that the *Optimized* method runs faster at higher probability thresholds, since more candidates are easier to disqualify by checking their probability upper bounds. As expected, the *Basic* method exhibits constant performance since it always evaluates the qualification probability for the entire interval, regardless of α.

Finally, we compare the processing time by varying the length of query interval (time or path). As shown in Figure 10d and Figure 11d, although both methods consume more CPU time when the query interval is longer, their performance gap increases remarkably.

5. RELATED WORK

Since the problem investigated in our paper is a marriage between uncertain moving objects and querying processing on road networks, we will review the related work in these two categories respectively in this section.

5.1 Managing Uncertainty of Moving Objects

Uncertainty issues in moving object databases have been addressed in several literatures before. Wolfson et al. [33][32] addressed the update problem in moving objects by proposing an information cost model that captures uncertainty, deviation and communication. The authors analysed *dead-reckoning* policies that update the database location whenever the distance between the actual location and the database location exceeds a given threshold. Be-

sides, they also considered the problem of range queries and proposed a probabilistic approach to solve it. Pfoser and Jensen [23] presented a formal quantitative approach to the aspect of uncertainty in modeling moving objects. They demonstrated that, under constraint maximum velocity, the spatial zone of the object's whereabouts during two consecutive sampling positions is an ellipse. But the authors limited the uncertainty to the past of the moving objects and the error may become very large as time approaches now. Uncertainty of moving objects was also treated by Sistla et al. [24], who introduced a data model, called MOST, to represent moving objects with uncertain positions, and proposed Future Temporal Logic (FTL) as the query language for the MOST model. By considering the measurement error when capturing the object movement, Trajcevski et al. propose the concept of *uncertain trajectory* in which the uncertainty of a trajectory is modelled by a three-dimensional sheared cylinder. Then based on this model, the authors introduced a set of spatio-temporal operators and proposed efficient algorithms for continuous range queries [29] and nearest neighbor queries [28]. A different model was proposed by Cheng et al. [5], in which the location uncertainties are updated at every time instant and range queries are issued at current time point also. However, their method is not efficient for continuous queries as a query must be re-evaluated at each time instant, resulting in high query costs. Recently, Zhang et al. [34] devised an efficient inference method for predicating future locations and integrated it into indexing structures designed for uncertain moving objects. But their assumption that the distribution of current locations and velocities of moving objects is known at any time may not be practical in real applications.

However, all the above work target Euclidean space where the distance between objects and queries is defined as a function of their coordinates. Hence their methods are not applicable to our problem settings, where the distance function depends on the connectivity and weights of the underlying road networks.

5.2 Query Processing on Road Networks

To process spatial queries on road networks, several algorithms have been developed using the network distance. Shahabi et al. [26] propose an embedding technique to transform a road network into a higher dimensional space in order to utilize computationally simple metrics. The main disadvantage of this method is that it provides only an approximation of the actual distance. Jensen et al. [14] formalize the problem of kNN search in road networks and present a system prototype for such queries. They use algorithms similar to Dijkstra's algorithm in order to perform online calculations of the shortest distance from a query point to an object. Shekhar et al. [27] present four alternative techniques for finding the first nearest neighbor to a moving query object on a given path. Papadias et al. [22] describe a framework that integrates network and Euclidean information, and answers kNN, range, closest pairs and e-distance join queries. They index the data objects with an R-tree and utilize connectivity and location information to guide the search. Kolahdouzan et al. [16] propose a solution-based approach to retrieve the kNNs based on pre-computed network Voronoi cells.

Efficient processing of continuous queries in road networks has also been studied recently. Kolahdouzan et al. [15] and Cho et al. [6] develop different techniques, UBA and UNICONS, to reduce the number of kNN evaluations by allowing the kNN result to be valid for a time interval. However, they are designed to handle queries over static data objects. Mouratidis et al. [21] address the issue of continuous monitoring kNNs over moving objects and propose an incremental monitoring algorithm to re-evaluate the query when updates occur.

The uncertainty issue in network-constraint moving objects is also considered in [7]. But it is different with our work in two folds. First, they assume the route between consecutive sampling positions is deterministic. Second, the main focus of their work is to define the algebra (i.e., data type, operator) to support uncertainty. More recently, [17] have adopted the space-time prism in the road-network context, however, no query processing aspects have been addressed. To the best of our knowledge, our work is the foremost one in comprehensive addressing of the probabilistic uncertainty model, its impact on the syntax of continuous queries, and the corresponding processing methods for range queries over uncertain objects on road networks.

6. CONCLUSION AND FUTURE WORK

We addressed the problem of efficient processing spatio-temporal range queries over uncertain trajectories in road networks. The main motivation of this work is that in many realistic settings, the positions of moving objects are sampled only at discrete time instants, and little can be known about their whereabouts in-between consecutive samples. Most of the work that have incorporated this uncertainty into the trajectory models have targeted the motion in Euclidean space, where there are no constraint on the movement of objects. Although [7, 17] have considered the problem of uncertainty in road-network settings, the impact of the model on the processing of spatio-temporal queries has not been addressed.

By considering the maximum speed on each road segment as the only restricting parameter, we quantitatively model the uncertain location of a moving object as a time-dependent *probability distribution function*. With this, we are able to formally define the static and continuous range queries whose syntax captures the impact of the uncertainty of the moving objects on road networks. To facilitate the efficient query processing, we proposed a novel indexing structure that incorporates the uncertainty of the spatial whereabouts. In addition, we developed effective filtering strategies and efficient algorithms for the refinement stage of the (continuous) range queries processing. As demonstrated by our experiments, our methods can significantly reduce both the I/O cost and CPU time.

An immediate future work is to address the *Continuous Probabilistic Nearest Neighbor* queries based on our model. Similar to the continuous range queries, one of the main challenges is to identify some *critical changing points*, in-between which certain important properties of the objects with respect to the query parameters are ensured. Another possible extension is to study the efficient processing of aggregate queries, e.g., estimate the number of moving objects inside some region within a given time interval, which have many applications in real transportation systems. Of course the major challenge still comes from the high evaluation cost given that the locations of moving objects are all probabilistic, for which approximate techniques may be a promising direction to pursue. Finally, we also plan to investigate the space-efficiency of the UTH index structure in our future work.

7. ACKNOWLEDGEMENTS

We wish to thank the anonymous reviewers for several comments and suggestions that have improved the paper. This work was supported by the ARC grants DP110103423, DP0987557, and NSF-CNS grant 0910952.

8. REFERENCES

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *SODA*, pages 589–598, 2003.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, 2005.

[3] J. Chen and R. Cheng. Efficient evaluation of imprecise location-dependent queries. In *ICDE*, pages 586–595, 2007.

[4] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, pages 551–562, 2003.

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

[6] H. Cho and C. Chung. An efficient and scalable approach to cnn queries in a road network. In *VLDB*, page 876, 2005.

[7] V. de Almeida and R. Güting. Supporting uncertainty in moving objects in network databases. In *ACM GIS*, page 40, 2005.

[8] H. Ding, G. Trajcevski, and P. Scheuermann. Towards efficient maintenance of continuous queries for trajcectories. *GeoInformatica*, 12(3), 2008.

[9] E. Frentzos. Indexing objects moving on fixed networks. *Advances in Spatial and Temporal Databases*, pages 289–305, 2003.

[10] B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10), 2006.

[11] J. Greenfeld. Matching gps observations to locations on a digital map. In *81th Annual Meeting of the Transportation Research Board*, 2002.

[12] R. Güting, V. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *VLDB J.*, 15(2), 2006.

[13] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[14] C. Jensen, J. Kolář, T. Pedersen, and I. Timko. Nearest neighbor queries in road networks. In *ACM GIS*, page 8, 2003.

[15] M. Kolahdouzan and C. Shahabi. Continuous k-nearest neighbor queries in spatial network databases. In *STDBM*, pages 33–40, 2004.

[16] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*, page 851, 2004.

[17] B. Kuijpers and W. Othman. Modelling uncertainty on road networks via space-time prisms. *Int.l Journal on GIS*, 23(9), 2009.

[18] B. Kuijpers and W. Othman. Trajectory databases: data models, uncertainty and complete query languages. *Journal of Computer and System Sciences*, 2009. doi:10.1016/j.jcss.2009.10.002.

[19] J. Lema, L. Forlizzi, R. Güting, E. Nardelli, and M. Schneider. Algorithms for moving objects databases. *The Computing Journal*, 46(6), 2003.

[20] M. F. Mokbel and W. G. Aref. Sole: scalable on-line execution of continuous queries on spatio-temporal data streams. *VLDB J.*, 17(5):971–995, 2008.

[21] K. Mouratidis, M. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, page 54, 2006.

[22] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, page 813, 2003.

[23] D. Pfoser and C. Jensen. Capturing the uncertainty of moving-object representations. In *SSD*, pages 111–131, 1999.

[24] A. Prasad Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Querying the uncertain position of moving objects. *Temporal Databases: Research and Practice*, pages 310–337, 1998.

[25] K.-F. Richter. A uniform handling of different landmark types in route directions. In *International Conference On Spatial Information Theory (COSIT)*, 2007.

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

[27] S. Shekhar and J. Yoo. Processing in-route nearest neighbor queries: a comparison of alternative approaches. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, pages 9–16, 2003.

[28] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *EDBT*, pages 874–885, 2009.

[29] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems (TODS)*, 29(3):507, 2004.

[30] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *EDBT*, pages 145–161, 2002.

[31] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms. In *SSDBM*, 2006.

[32] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *ICDE*, pages 588–596, 2002.

[33] O. Wolfson, A. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7(3):257–387, 1999.

[34] M. Zhang, S. Chen, C. Jensen, B. C. Ooi, and Z. Zhang. Effectively indexing uncertain moving objects for predictive queries. In *VLDB*, pages 261–272, 2009.