Constructing Popular Routes from Uncertain Trajectories

Ling-Yin Wei
National Chiao Tung University
Hsinchu, Taiwan
lywei.cs95g@nctu.edu.tw

Yu Zheng
Microsoft Research Asia
Beijing, China
yuzheng@microsoft.com

Wen-Chih Peng
National Chiao Tung University
Hsinchu, Taiwan
wcpeng@cs.nctu.edu.tw

ABSTRACT

The advances in location-acquisition technologies have led to a myriad of spatial trajectories. These trajectories are usually generated at a low or an irregular frequency due to applications' characteristics or energy saving, leaving the routes between two consecutive points of a single trajectory uncertain (called an uncertain trajectory). In this paper, we present a Route Inference framework based on Collective Knowledge (abbreviated as RICK) to construct the popular routes from uncertain trajectories. Explicitly, given a location sequence and a time span, the RICK is able to construct the top-*k* routes which sequentially pass through the locations within the specified time span, by aggregating such uncertain trajectories in a mutual reinforcement way (i.e., uncertain + uncertain → certain). Our work can benefit trip planning, traffic management, and animal movement studies. The RICK comprises two components: *routable graph construction* and *route inference*. First, we explore the spatial and temporal characteristics of uncertain trajectories and construct a routable graph by collaborative learning among the uncertain trajectories. Second, in light of the routable graph, we propose a routing algorithm to construct the top-*k* routes according to a user-specified query. We have conducted extensive experiments on two real datasets, consisting of Foursquare check-in datasets and taxi trajectories. The results show that RICK is both effective and efficient.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *data mining, spatial databases and GIS.*

General Terms

Algorithms, Design, Experimentation.

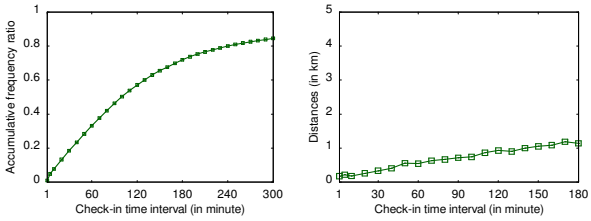
Keywords

Collaborative learning, trajectory data mining, route inference, social media.

1. INTRODUCTION

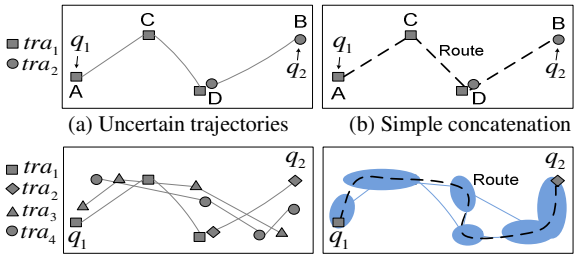
The increasing availability of location-acquisition technology (e.g., GPS), has led to a huge volume of spatial trajectories that represent the movement routes of humans, animals, hurricanes, and vehicles. Without loss of generality, a trajectory is a sequence of data points where each data point records location information and a time-stamp [18]. For example, the driving routes of vehicles and migratory routes of animals are usually recorded by GPS trajectories. Meanwhile, users could perform check-in services (e.g., Foursquare) to note their locations via a mobile phone and share their photos and activities. The time-ordered check-in records of a user are able to be expressed by trajectories. Moreover, on a photo sharing website (e.g., Flickr), people share geotagged photos whose time-stamps and geolocations can be represented as trajectories as well. However, these trajectories are usually generated at a low frequency due to energy saving and features of applications, resulting in the uncertainty of a moving object's mobility in a trajectory.

Figure 1 shows statistic information from Foursquare datasets in Manhattan. As shown in Figure 1(a), most check-in time intervals vary from 1 to 180 minutes. Moreover, we further investigate the distances among these check-in records. The medians of the distances between two check-in records are less than two kilometers in Figure 1(b). The above two observations show that even in Manhattan, which has a lot of tourists, the uncertain routes apparently exist between two check-in records.



(a) (b)

Figure 1. Observations from Foursquare datasets.



(a) Uncertain trajectories (b) Simple concatenation

(c) Uncertain trajectories (d) Mutual reinforcement

Figure 2. Examples of uncertain trajectories.

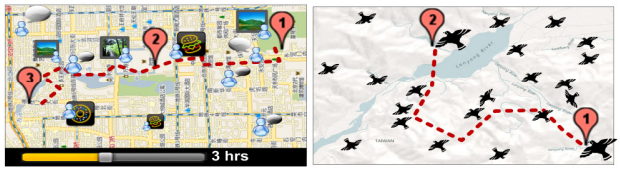
These low sampled trajectories do not detail the routes, and raise uncertain routes between two consecutive sampled points in the trajectories. In this paper, we call such trajectories uncertain trajectories. Examples of uncertain trajectories are illustrated in Figure 2. Figure 2(a) shows two check-in trajectories, tra_1: A → C → D and tra_2: D → B, in a rural space (i.e., road network information is not available). If a tourist would like to travel from q_1 to q_2, he/she may have no idea of how to travel without the aid of road networks or by referring to a trajectory (e.g., tra_1 or tra_2). In addition, given one migratory trajectory of a bird, we do not know where the bird flew between two sampled points which are

KDD'12, August 12–16, 2012, Beijing, China.

several miles away from each other. Due to the uncertainty of low sampled trajectories, how to derive detailed routes from uncertain trajectories is an important task.

The prior work [16] proposed a framework to discover the routes from historical trajectories. Explicitly, given a set of historical trajectories, an underlying road network, and a location sequence, the work aims to suggest the top-k possible routes sequentially passing the queried locations. Note that by the aid of the given road network, the work explores possible routes derived from road networks. However, for some applications (e.g., animal migration routes or hurricane routes), road network information is not available. As for check-in datasets and geo-photo datasets, the service providers may not have road network information either. Without road network information, the work [16] cannot derive the top-k routes.

In this paper, without road network information, we propose a Route Inference framework based on Collective Knowledge (abbreviated as RICK) to construct the popular routes from uncertain trajectories. Explicitly, given a location sequence and a time span, RICK constructs the top-k routes, sequentially passing the locations within the specified time span. The RICK is beneficial for many practical applications. Examples of applications are trip planning [2,5,7,8,15,17], animal movement behavior studies [9] and traffic flow analysis [14]. For example, a user plans to take a tour that consists of three attractions (e.g., the Temple of Heaven, the Palace Museum and the Houhai Bar Street in Figure 3(a)) in Beijing, while having no idea of how to travel around them. At this moment, the RICK can recommend the popular travel route, inferred from check-in records (or geo-tagged photos) generated by other tourists. Another example in Figure 3(b) is to help biologists discover birds' moving behaviors from uncertain trajectories.



(a) (b)

Figure 3. Scenarios of applications.

To infer routes from uncertain trajectories, in many cases, we need to construct a route based on segments from multiple uncertain trajectories as there is no historical trajectory passing all the queried locations. For instance, in Figure 2(a), tra_1 and tra_2 do not pass through both q_1 and q_2. If a tourist would like to travel from q_1 to q_2, existing trip planning [1,7,15,17] could explore the sequential relations among these places and derive A → C → D → B by concatenation in Figure 2(b). However, the distances between two consecutive locations are still far away and a travel route still cannot be derived according to the users' trajectory. In addition, one could extract the trajectories that capture similar movements. However, due to the sparseness of data points in uncertain trajectories, two uncertain trajectories usually have totally different geospatial representations even though the two trajectories follow the same route or have the same subroutes (i.e., are correlated). As such, the similarity between two uncertain trajectories is hard to measure.

In this paper, given a set of uncertain trajectories (e.g., Figure 2(c)), a routable graph (e.g., the blue part in Figure 2(d)) is generated for indicating routing information in a free space by exploring spatio-temporally correlated uncertain trajectories. In light of the routable graph, we have designed a route score function and proposed a routing algorithm to construct the top-k routes (e.g., the dashed curve in Figure 2(d)) satisfying the query. We have conducted extensive experiments on two real datasets and the results show the effectiveness and efficiency of the RICK.

The contributions of this paper are summarized as follows:

- Without the aid of road networks, we develop a route inference framework to infer routes from uncertain trajectories.
- We propose a routable graph with routing information, generated by exploring spatio-temporal correlations among uncertain trajectories.
- In light of the routable graph, we define a route score function and develop a routing algorithm to construct the top-k routes.
- We have conducted extensive experiments using real datasets of 15,000 driving trajectories and 6,600 check-in sequences. The results indicate the RICK is effective and efficient.

The remainder of the paper is organized as follows. Section 2 gives the preliminary of our work. Section 3 illustrates the routable graph construction. Section 4 details the route inference. Section 5 presents the experimental results. Section 6 reviews related work. Section 7 concludes the paper.

2. PRELIMINARY

We present some terms and the problem addressed in this paper, and then overview the proposed framework.

Definition 1 (*Trajectory*): A *trajectory* tra_i is a time-ordered sequence of sampled points, i.e., $tra_i: p_1^i \rightarrow p_2^i \rightarrow \cdots \rightarrow p_n^i$. Each point $p_k^i \in tra_i$ is represented by $(p_k^i.l, p_k^i.t)$ where $p_k^i.l$ is a geographic coordinate (a location for short) at a time-stamp $p_k^i.t$.

As discussed before, trajectories are usually generated at a low sampling rate, leading to the real route between two consecutive points of a trajectory being uncertain. If a time interval between two consecutive sampled points is large, the uncertainty of the route between the two points would increase. In this paper, we further claim that road networks are not always available for inferring routes between two locations. For animal trace data and outdoor activities in urban areas, the movements are not along road networks. Thus, our problem is defined as follows:

***Problem*:** Given an uncertain trajectory dataset D and a user-specified query consisting of a time span Δt and a location sequence $q: q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_m$, we infer the top-$k$ popular routes in a free space such that each route sequentially traverses the given locations, and the travel time of the route between any two consecutive query locations is within Δt.

Figure 4 overviews our framework RICK, which consists of two components: routable graph construction and route inference.

***Routable graph construction*:** This offline component builds up a routable graph from an uncertain trajectory dataset. To generate a routable graph, there are two stages: region construction and edge inference. First, we partition the space into disjoint cells and then index the given uncertain trajectories in the gridded space. By exploring the spatio-temporal characteristics of the uncertain trajectories passing these cells, we merge these individual cells to form some geographical regions. Here, each cell forms a vertex in the routable graph that we are going to build up. Second, we infer the edges between the cells with the uncertain trajectories. These edges can be categorized into two types: they are inside a region or they are between regions. The information inferred for an edge comprises a moving direction, a transition support, and a travel time, indicating the transition relationship between two cells.

***Route inference*:** This component, consisting of the route generation and the route refinement, is responsible for on-line queries. In the route generation stage, given a query, we propose a routing algorithm to infer the top-k rough routes, each of which is represented by a sequence of cells, with the constructed graph. The routing algorithm first finds out the qualified subroutes between any two consecutive query locations, and then concatenates these subroutes into completed routes in a branch-and-bound manner. In addition, we define a score function based on historical movements for ranking these routes. In the route refinement stage, we further refine each rough route to derive a detailed route represented by a sequence of consecutive segments from historical data points of uncertain trajectories.

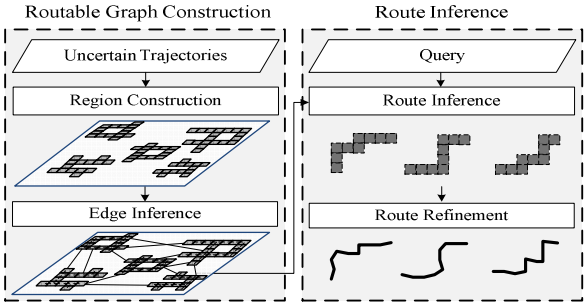


Figure 4. Overview of RICK.

Table 1. Notations

Symbol	Description
p_j^i	The jth point in trajectory i
$p.g$	The cell that p locates in
(x,y)	A cell id
c(x,y)	The number of distinct trajectories traversing (x,y)
θ	A temporal constraint
C	A minimum connection support
$\mathcal{T}(g \to g', tra)$	A set of the travel times from g to g' by tra

3. ROUTABLE GRAPH CONSTRUCTION

3.1 Region Construction

To construct a routable graph, we discover connected geographical areas by collaborative learning among historical uncertain trajectories. We first observe the spatial and temporal characteristics of the uncertain trajectories. For instance, Figure 5(a) shows three trajectories, $tra_1: p_1^1 \xrightarrow[10\ mins]{} p_2^1 \xrightarrow[20\ mins]{} p_3^1$, $tra_2: p_1^2 \xrightarrow[13\ mins]{} p_2^2 \xrightarrow[10\ mins]{} p_3^2$, and $tra_3: p_1^3 \xrightarrow[30\ mins]{} p_2^3$, where times are travel times between two consecutive points. The locations of data points of tra_1 and tra_2 are different even if the two trajectories follow the same route (e.g., the black solid line). We observe that 1) $p_2^1.l$ and $p_2^2.l$ are at the same place; 2) tra_1 and tra_2 have similar travel times from their first points (e.g., p_1^1, p_1^2) to the place; 3) $p_1^1.l$ and $p_1^2.l$ are spatially close. The observations indicate that the route of tra_1 from p_1^1 to p_2^1 and the route of tra_2 from p_1^2 to p_2^2 may be the same. We say that the two subtrajectories $p_1^1 \to p_2^1$ and $p_1^2 \to p_2^2$ are *spatio-temporally correlated (st-correlated)*. Moreover, if p_1^1 and p_1^2 are sampled on the same route, $p_1^1.l$ and $p_1^2.l$ could be connected. Specifically, this means that there exists at least one route between $p_1^1.l$ and $p_1^2.l$. On the other hand, in Figure 5(a), although $p_2^1.l$ and $p_2^3.l$ are at the same place and p_1^1 and p_1^3 are spatially close, tra_3 may be sampled from the other route (e.g, the green dotted line). The reason is that tra_3 has a longer travel time from p_1^3 to p_2^3.

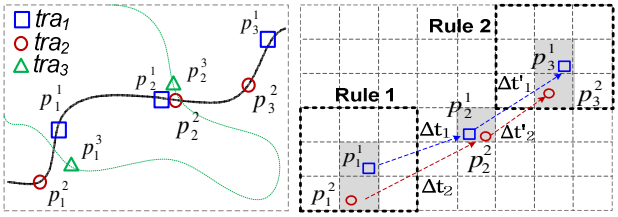


Figure 5. Spatio-temporally correlated uncertain trajectories.

Based on the aforementioned observations, we define some terms for constructing connected geographical areas. Table 1 summarizes the notations used in this paper. To clearly describe spatial relations among data points of uncertain trajectories, we adopt a gridded space. First, we divide a geographical range into disjoint cells by a given cell length l. The set of the cells is denoted as $\mathcal{G}$, and the GID of a cell g is represented by (x, y). Moreover, each point of an uncertain trajectory can be mapped into a cell, and an uncertain trajectory can be transformed into a sequence of cells. As such, given an uncertain trajectory $tra_i: p_1^i \to p_2^i \to \cdots \to p_n^i$, the trajectory can be transformed into $p_1^i.g \to p_2^i.g \to \cdots \to p_n^i.g$, where $p_k^i.g$ represents the cell that $p_k^i.l$ locates in. Given two cells $g = (x, y)$ and $g' = (x', y')$, the cells g and g' are called *spatially close* if $|x - x'| \le 1$ and $|y - y'| \le 1$.

To explore connected geographical areas, we formally define st-correlated relations among uncertain trajectories. Given a cell g, if $g = p_k^i.g$ for some $p_k^i \in tra_i$, we say that tra_i traverses cell g, denoted $g \in tra_i$. If tra_i traverses from cell g to cell g', we say $g \to g' \in tra_i$. The set of the travel times of $g \to g'$ by tra_i is denoted by $\mathcal{T}(g \to g', tra_i)$. If $g \to g' \in tra_i$, $\mathcal{T}(g \to g', tra_i) \neq \emptyset$.

Definition 2 (*Spatio-temporally correlated relation*): Given two uncertain trajectories $tra_i: p_1^i \to \cdots \to p_n^i$ and $tra_j: p_1^j \to \cdots \to p_m^j$, and *a temporal constraint* θ, tra_i's subtrajectory $p_k^i \to \cdots \to p_{k'}^i$ and tra_j's subtrajectory $p_h^j \to \cdots \to p_{h'}^j$ are st-correlated if

1) $\exists\, \Delta t_1 \in \mathcal{T}(p_k^i.g \to p_{k'}^i.g, tra_i), \Delta t_2 \in \mathcal{T}(p_h^j.g \to p_{h'}^j.g, tra_j)$ s.t. $\frac{|\Delta t_1 - \Delta t_2|}{\max\{\Delta t_1, \Delta t_2\}} \le \theta$;

2) one of the two rules is satisfied:

Rule1: $p_k^i.g$ and $p_h^j.g$ are spatially close, and $p_{k'}^i.g = p_{h'}^j.g$.

Rule2: $p_k^i.g = p_h^j.g$, and $p_{k'}^i.g$ and $p_{h'}^j.g$ are spatially close.

Note that if a trajectory tra' is a subtrajectory of a trajectory tra, we denote it as $tra' \subseteq tra$.

Given trajectories in Figure 5(a), Figure 5(b) depicts the st-correlated relation between tra_1 and tra_2. Let $\Delta t_1 \in \mathcal{T}(p_1^1.g \to p_2^1.g, tra_1)$ and $\Delta t_2 \in \mathcal{T}(p_1^2.g \to p_2^2.g, tra_2)$ and assume Δt_1 and Δt_2 satisfy a given temporal constraint. According to Rule 1 in Definition 2, $p_1^1 \to p_2^1$ and $p_1^2 \to p_2^2$ are st-correlated, because $p_1^1.g$ and $p_1^2.g$ are spatially close and $p_2^1.g = p_2^2.g$. Similarly, let $\Delta t'_1 \in \mathcal{T}(p_2^1.g \to p_3^1.g, tra_1)$ and $\Delta t'_2 \in \mathcal{T}(p_2^2.g \to p_3^2.g, tra_2)$ and assume $\Delta t'_1$ and $\Delta t'_2$ satisfy a given temporal constraint. According to Rule 2 in Definition 2, $p_2^1 \to p_3^1$ and $p_2^2 \to p_3^2$ are st-correlated since $p_2^1.g = p_2^2.g$ and $p_3^1.g$ and $p_3^2.g$ are spatially close.

Definition 3 (*Connection support*): Given an uncertain trajectory dataset D, a set of cells $\mathcal{G}$, a temporal constraint θ, and two cells $g, g' \in \mathcal{G}$, where g and g' are spatially close, *the connection support of the cell pair* (g, g') is defined as $|T_1 \cup T_2|$ where $T_1 = \{(tra_i, tra_j) | tra_i'$ and tra_j' are st-correlated, $g \to g'' \in tra_i'$, and $g' \to g'' \in tra_j'$ for some $g'' \in \mathcal{G} - \{g, g'\}$, $tra_i' \subseteq$

$tra_i, tra_j' \subseteq tra_j$, and T_2 ={(tra_i, tra_j)| tra_i' and tra_j' are st-correlated, $g'' \rightarrow g \in tra_i'$, and $g'' \rightarrow g' \in tra_j'$ for some $g'' \in \mathcal{G} - \{g, g'\}, tra_i' \subseteq tra_i, tra_j' \subseteq tra_j\}$.

For example, in Figure 5(b), given $p_1^1.g$ and $p_1^2.g$, which are spatially close, the support of the cell pair $(p_1^1.g, p_1^2.g) = |T_1 \cup T_2| = 1$ because $T_1 = \{(tra_1, tra_2)\}$ and $T_2 = \emptyset$. Similarly, given $p_2^1.g$ and $p_3^2.g$, which are spatially close, the support of the cell pair $(p_3^1.g, p_3^2.g) = |T_1 \cup T_2| = 1$ because $T_1 = \emptyset$ and $T_2 = \{(tra_1, tra_2)\}$.

Definition 4 (*Neighbor*): Given an uncertain trajectory dataset *D*, a set of cells $\mathcal{G}$, two cells $g, g' \in \mathcal{G}$, a temporal constraint θ, and a *minimum connection support C*, if the connection support of the cell pair (g , g') is greater than or equal to *C*, g and g' are *neighbors*, denoted as gNg'.

We define a *region* as a connected geographical area as follows:

Definition 5 (*Region*): Given a set of cells $\mathcal{G}'$, $\mathcal{G}'$ forms *a* region if for any two cells $g, g' \in \mathcal{G}'$, there exists a chain of cells $(g =) g_1 = g_2 = \cdots = g_k (= g')$ s.t. $g_i N g_{i+1}$ for each $g_i \in \mathcal{G}'$ and $i \in [1, k)$.

To construct regions, a naïve method is that we generate all cell pairs from the set of cells $\mathcal{G}$ and then compute the connection support of each cell pair by checking other cells in $\mathcal{G}$. We then verify whether the connection supports of cell pairs satisfy the given minimum connection support *C* to construct regions. However, the time complexity of the method is costly. In this paper, we propose an efficient algorithm to construct regions.

The proposed algorithm utilizes an index structure presented as follows. Given a cell length and an uncertain trajectory dataset *D*, we build up a *grid index* in which each cell g has a unique GID, a value $c(g)$, and a corresponding trajectory list. Note that $c(g) = |\{tra | g \in tra, tra \in D\}|$. In the grid index, each GID indexes a list of trajectories that records which uncertain trajectories traverse the cell and which points of these uncertain trajectories locate in the cell by TIDs and PIDs, respectively. To improve the efficiency of the region construction, the trajectories in a cell's corresponding trajectory list are sorted by m_{tra}, where m_{tra} is the median of $\{c(p.g) | p \in tra\}$ with given tra.

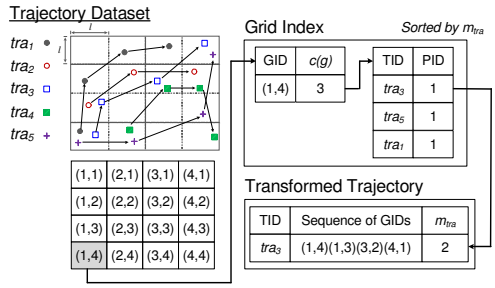


Figure 6. An example of an index structure.

For instance, given a cell length l and an uncertain trajectory dataset $D = \{tra_1, tra_2, tra_3, tra_4, tra_5\}$, Figure 6 shows an example of an index structure. Given a cell (1,4), $c(1,4)=3$ since three distinct uncertain trajectories (i.e., tra_1, tra_3, tra_5) traverse cell (1,4). The trajectory list of cell (1,4) records these TIDs (i.e., tra_1, tra_3, tra_5) and the corresponding PID of each trajectory. The corresponding PID of tra_1 is 1 since the point of tra_1 that locates in cell (1,4) is the first point of tra_1. As shown in Figure 6, tra_3 traverses four cells (i.e., (1,4), (1,3), (3,2), and (4,1)), and we can calculate that $c(1,4)=3$, $c(1,3)=2$, $c(3,2)=2$, and $c(4,1)=2$. The median among {2, 2, 2, 3} is 2 and thus $m_{tra_3}=2$.

Before constructing regions, we let $\mathcal{G}=\mathcal{G} - \mathcal{G}'$, where $\mathcal{G}'=\{g | c(g) = 0, g \in \mathcal{G}\}$. The algorithm of region construction is detailed in Algorithm 1. Note that the term *enclosed* is defined as follows.

Algorithm 1: Region Construction

Input: An uncertain trajectory dataset *D*, a set of cells $\mathcal{G}$, a temporal constraint θ, and a minimum connection support *C*.
Output: A set of regions *R*.
1. $\mathcal{G}' \leftarrow$ Sort cells in $\mathcal{G}$ in a decreasing order of $c(g)$;
2. **Do**
3. $g \leftarrow$ Pop the cell from $\mathcal{G}'$;
4. **Foreach** *tra* traversing g by the order stored in the grid index
5. $\tau(g) \leftarrow \{p | p.g = g \, and \, p \in tra\}$;
6. **Foreach** $p \in tra - \tau(g)$ and $p.g$ is not enclosed
7. **If** $p.g$ is contained in some region
8. $r \leftarrow$ The region contains $p.g$;
9. **Else**
10. $r \leftarrow \emptyset$;
11. **If** p is before p' for all $p' \in \tau(g)$
12. $r \leftarrow \mathrm{CM}(r, p.g, g, \theta, C, Rule1)$;
13. **ElseIf** p is after p' for all $p' \in \tau(g)$
14. $r \leftarrow \mathrm{CM}(r, p.g, g, \theta, C, Rule2)$;
15. **Else**
16. $r \leftarrow \mathrm{CM}(r, p.g, g, \theta, C, Rule1)$;
17. $r \leftarrow \mathrm{CM}(r, p.g, g, \theta, C, Rule2)$;
18. $R \leftarrow R \cup \{r\}$;
19. **Until** $\mathcal{G}'$ is empty or each cell in $\mathcal{G}$ is in some $r \in R$;
20. **Return** *R*;

Algorithm 2: Cell Merging (CM)

Input: A region *r*, a cell *p.g*, a cell *g*, a temporal constraint θ, a minimum connection support *C*, and an indicator *I*
Output: A region *r*
1. Let $p.g = (x, y)$;
2. **Foreach** cell $g' = (x', y') \in \mathcal{G} - r$ where $|x' - x| \leq 1$ and $|y' - y| \leq 1$
3. **If** *I* is Rule1
4. Verify whether $p.gNg'$ is held with given g by rule 1;
5. **Else**
6. Verify whether $p.gNg'$ is held with given g by rule 2;
7. **If** $p.gNg'$
8. **If** $r = \emptyset$
9. $r \leftarrow \{p.g\}$;
10. **If** g' is contained in some region
11. $r' \leftarrow$ The region contains g';
12. **Else**
13. $r' \leftarrow \emptyset$
14. $r' \leftarrow \mathrm{CM}(r', g', g, \theta, C, I)$;
15. $r \leftarrow r \cup r'$;
16. **Return** *r*;

Definition 6 (*Enclosed*): Given a set of cells $\mathcal{G}$ and a cell $g \in \mathcal{G}$, the cell g is *enclosed* if there exists a region $r \subseteq \mathcal{G}$ s.t. $g, g' \in r$, $\forall g' \in \{g' | g'$ and g are spatially close, $g' \in \mathcal{G}\}$.

In Algorithm 1, we iteratively merge cells to form regions by calculating the connection supports of cell pairs. To efficiently construct regions, we determine an order for the calculation of connection supports of cell pairs according to $c(g)$. Once a cell is chosen, we iteratively pick a trajectory from the cell's trajectory list in Step 4. We then calculate the connection supports of the cell pairs around the points of the trajectory and merge qualified cells from Step 5 to Step 18. An example of this procedure is illustrated in Figure 7. Let a chosen cell be g. Assume the trajectory tra_1 traversed it and $p_2^1.g = p_4^1.g = g$ (i.e., $\tau(g) = \{p_2^1, p_4^1\}$). In Step 6, we pick a point (e.g., p_1^1) from tra_1 but not the point is not in $\tau(g)$. If the cell that the point locates in is not enclosed, the cell would be possibly merged with other cells. The connection supports of the cell pairs of $p_1^1.g$ and each cell around $p_1^1.g$ are calculated and the qualified cell pairs will be merged. Based on

$p_1^1.g$, a region (e.g., the blue cells in Figure 7) is generated in the first round and more cells (e.g., the red cells in Figure 7) are merged into the region around $p_1^1.g$ in the second round (i.e., Step 14 in Algorithm 2). Similarly, for $p_1^1.g$, a merging process will be stopped if no cell can be merged around $p_1^1.g$. We then chose other points (e.g., p_3^1, p_5^1) to construct regions around these points in the same way.

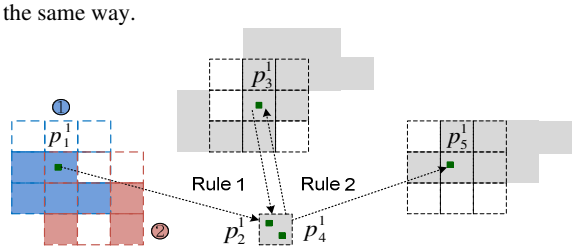


Figure 7. Region construction process.

Time complexity analysis: Given an uncertain trajectory dataset D and a set of cells $\mathcal{G}$, the time complexity of the naïve method is $O(mn^3)$ where $|\mathcal{G}| = n$ and $|D| = m$. Similarly, the time complexity of Algorithm 1 is $O(n(logn + cm^2))$, where c is the minimal number of the first loop. For Step 1 in Algorithm 1, it costs $O(nlogn)$ to sort cells in decreasing order of $c(g)$. In addition, the time complexity of Algorithm 2 is $O(m)$ because there are at most m uncertain trajectories for counting the connection support of a cell pair. Thus the time complexity of the remaining steps in Algorithm 1 is $O(cnm^2)$.

3.2 Edge Inference

Once the regions are generated, we then infer edges and derive edge information including moving directions, transition supports, and travel times from historical uncertain trajectories. To generate the edges of a routable graph, we infer the edges within each region, and then infer the edges among regions.

A routable graph is a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. Each vertex represents a geographical area, i.e., a cell. Each directed edge e indicates a transition relationship and has two attributes, *the transition support* $e.s$ and *the travel time* $e.t$. To derive the transition support of an edge, we record which distinct uncertain trajectories traverse the edge. In other words, an edge has an uncertain trajectory list to record which distinct uncertain trajectories traverse it.

According to the definition of a region, a region is composed of connected cells, and thus we first generate virtual bidirected edges between cells if the cells are neighbors in a region, To infer edges' realistic directions, transition supports, and travel times, we propose *a shortest path based inference approach*.

Given a region and an uncertain trajectory dataset, we utilize the uncertain trajectories traversing the region to derive edge information in the region. For each trajectory traversing the region, we infer the shortest path between any two consecutive points of the trajectory by virtual bidirected edges in the region. We illustrate edge inference in a region in Figure 8. As shown in Figure 8(a), four uncertain trajectories pass through the region. For instance, in Figure 8(b), an uncertain trajectory $p_1 \rightarrow p_2 \rightarrow p_3$ (blue squares) traverses the region, and we infer the shortest paths from p_1 to p_2 and the shortest paths from p_2 to p_3. As shown in Figure 8(b), we find one shortest path from p_1 to p_2, and two shortest paths from p_2 to p_3. After finding the shortest path between two consecutive locations, we divide the travel time evenly and add it to the travel time list of each edge in the shortest path. In addition, each edge of the shortest path adds the trajectory ID into its corresponding trajectory list, and the transition support of each edge in the shortest path is accumulated one. If there are multiple shortest paths between two consecutive locations, we similarly update the information of each edge in these paths.

By using historical uncertain trajectories to infer edge information in a region, we further eliminate the redundant edges in the region and the edges whose transition supports are 0. Given an edge e_1 from the cell g and the cell g', where g and g' are spatially close, the edge e_1 is *redundant* in a region if there exist an edge e_2 from g to g'' and an edge e_3 from g'' to g', such that 1) g and g'' are spatially close and g' and g'' are spatially close, 2) $\frac{e_2.s+e_3.s}{2} > e_1.s$, and 3) $\frac{|e_2.t+e_3.t-e_1.t|}{\max\{(e_2.t+e_3.t),e_1.t\}} \leq \theta$ where θ is a given temporal constraint. Figure 8(c) shows the inferred edges in the region after reducing edges. The travel time of each edge is estimated by the median of all the travel times of the edge.

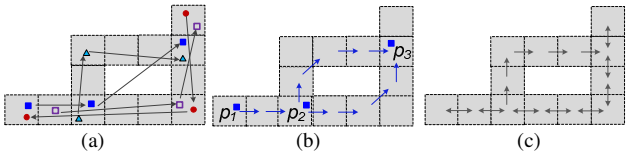


Figure 8. Edge inference in a region.

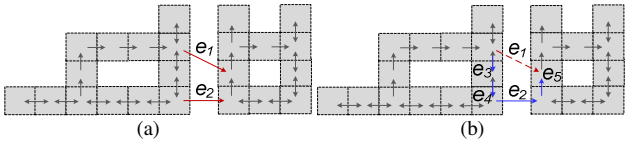


Figure 9. Edge inference between regions.

In the following, we construct edges between regions. Similarly, we generate edges between regions by using historical uncertain trajectories first. This means that if an uncertain trajectory traverses a cell of one region to a cell of another region, an edge is constructed between the two cells. Next, we eliminate the redundant edges between regions. Given an edge e_1 from the cell g to the cell g', where g and g' are in different regions, the edge e is a *redundant* edge between the two regions if there exists an alternative route $e_i \rightarrow \cdots \rightarrow e_j$ from the cell g to the cell g' such that 1) $\frac{1}{j-i+1}\sum_{k=i}^{j} e_k.s > e.s$, and 2) $\frac{\left|\sum_{k=i}^{j} e_k.t - e.t\right|}{\max\left\{\sum_{k=i}^{j} e_k.t, e.t\right\}} \leq \theta$ where θ is a given temporal constraint. Figure 9 shows an example of edge inference between two regions. As shown in Figure 9(a), edge e_1 and edge e_2 are generated between the two regions by historical uncertain trajectories. For instance, in Figure 9(b), edge e_1 is a redundant edge if 1) $\frac{1}{4}\sum_{k=2}^{5} e_k.s > e_1.s$, and 2) $\frac{|\sum_{k=2}^{5} e_k.t - e_1.t|}{\max\{\sum_{k=2}^{5} e_k.t, e_1.t\}} \leq \theta$ with a given θ.

Note that the transition information of an eliminated edge is propagated to alternative routes. The travel time of an eliminated edge is evenly propagated to the edges of each alternative route. The trajectory list of an eliminated edge is updated to each edge's corresponding trajectory list in alternative routes.

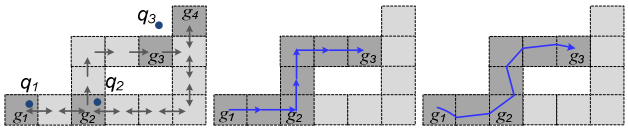
4. ROUTE INFERENCE

Given a location sequence and a time span, we generate the top-k popular routes by two phases: route generation and route refinement. In the first phase, we propose routing algorithms to search for the top-k coarse routes with the routable graph. We further refine the discovered the top-k routes to effectively derive specific routes in the second phase.

In the route generation phase: we first generate possible routes between each two consecutive queried locations (called *local routes*) and then search for the top-*k* routes (called *global routes*) from the generated local routes.

A *route* derived in this phase is represented by a sequence of vertices with a given graph $G = (V, E)$. Note that a vertex in the graph represents a cell; thus a route here is regarded as a sequence of cells, denoted as p: $g_1 \longrightarrow g_2 \longrightarrow \cdots \longrightarrow g_k$. Given a sequence of query locations and a time span, we search for qualified routes between any two consecutive query locations with the constructed graph. Before searching for routes with the graph, we need to specify the corresponding vertices of query locations. Since a vertex represents a geographical area of a cell, a query location can be mapped to the vertex whose corresponding geographical area overlaps the location. However, it is possible that a query location cannot be mapped to any vertex in the graph. We further select the vertices whose corresponding cells are close to the query location. We adopt the minimum distance (MINDIST) [12] to formulate the distance between a query location and a cell. According to the distance measurement, we specify the cells that are close to such query locations. Thus, a given location sequence is transformed into a sequence of sets of cells. Moreover, we transform a location sequence into cell sequences by combining these cells. After query transformation, we search for the top-*k* routes according to each cell sequence.

For instance, given a location sequence q: $q_1 \rightarrow q_2 \rightarrow q_3$ in Figure 10(a), the locations q_1 and q_2 are mapped to cells g_1 and g_2, respectively. By the minimum distance measurement, the set of cells $\{g_3, g_4\}$ is used to represent the location q_3. Then, the location sequence $q_1 \rightarrow q_2 \rightarrow q_3$ is transformed into two cell sequences, i.e., $g_1 \rightarrow g_2 \rightarrow g_3$ and $g_1 \rightarrow g_2 \rightarrow g_4$.



(a)Query transformation. (b)Route generation. (c)Route refinement

Figure 10. Route inference.

We generate routes with respect to each cell sequence. Before introducing the routing algorithm, we define the score function for the routes as follows.

Definition 7 (*Route score*): Given a graph $G = (V, E)$, a route $\mathcal{p}$: $\mathcal{p}_1 \rightarrow \mathcal{p}_2 \rightarrow \cdots \rightarrow \mathcal{p}_m$, where $\mathcal{p}_i$: $g_{i_1} \longrightarrow g_{i_2} \longrightarrow \cdots \longrightarrow g_{i_j}$, the score of the route is defined as $f(\mathcal{p}) = \sum_{i=1}^{m} \rho(\mathcal{p}_i)$, where $\rho(\mathcal{p}_i) = \frac{1}{j-1} |\cup_{k=1}^{j-1} \{tra | g_{i_k} \rightarrow g_{i_{k+1}} \in tra\}|$.

For each cell sequence, we first search for the top-*k* local routes between any two consecutive cells in the cell sequence (e.g., *g* and *g* ') by an A*-like routing algorithm. However, a possible maximum speed could be derived from historical uncertain trajectories or be determined by difference applications. Given a maximum speed, possible positions between any two consecutive query locations can be restricted in a range if a time interval between the two locations is specified [11]. That is, the possible routes are restricted in the cells overlapping the range. For the A*-like routing algorithm, an estimated score of a route from a cell *g* to a cell *g*' is represented as follows.

Given two cells *g* and *g*', a current visited cell *g*'', and a specified range *r*, an estimated score of a route $\mathcal{p}$: $\mathcal{p}_c \longrightarrow \mathcal{p}_f$ from a cell *g* to a cell *g*' in a specified range *r* is $\hat{f}(\mathcal{p}) = \rho(\mathcal{p}_c) + h(\mathcal{p}_f)$, where $\mathcal{p}_c$ is a known route from cell *g* to cell *g*'', and $h(\mathcal{p}_f)$ is the score of an estimated route $\mathcal{p}_f$ from cell *g*'' to cell *g*'.

Definition 8 (*Optimal score*): Given a graph $G = (V, E)$, an uncertain trajectory dataset *D*, a specified range *r*, and two cells *g* and *g*', the optimal score of the routes from a cell *g* to a cell *g*' in *r* is defined as

$$\hat{h}(\mathcal{p}) = |\{tra | tra \text{ passes through the range } r, tra \in D\}|$$

for some estimated route $\mathcal{p}$ from a cell *g* to a cell *g*'.

Once a local route is generated from the cell *g* to the cell *g*' and satisfies the given time span, the score of the local route is calculated. If there are more than *k* local routes constructed from the cell *g* to the cell *g*', the *k*-th maximum score of these local routes is recorded and incrementally updated. Based on the estimated optimal score function, a branch of searching routes will be stopped if the optimal score of routes generated from the branch is less than the updated *k*-th maximum score.

Based on the top-*k* local routes between any two consecutive cells of each cell sequence, we search for the top-*k* global routes by a branch-and-bound search approach. For instance, given a cell sequence $g_1 \rightarrow g_2 \rightarrow g_3$ in Figure 10(a), a global route is derived as a sequence of cells (dark grey) in Figure 10(b). To derive a specific route, a route is further transformed into a line by concatenating the centers of any two consecutive cells in the route. Figure 10(b) shows an example of such a route by a blue line.

However, it is possible that we search for local routes between two cells belonging to different regions. In the A*-like algorithm, although we search for local routes between two given cells in a restricted range of a graph, the search space is still large if the distance between two given cells is far and they are in different regions. It induces that a route between the two cells would possibly pass through several other regions. On the other hand, a lower bound of transition times between any two regions can be estimated by edge information. It helps us stop searching for routes between two regions if the lower bound of transition time between the two regions exceeds the time span. Hence, to improve the efficiency of route generation, we modify the proposed A*-like routing algorithm and introduce a two-layer routing algorithm.

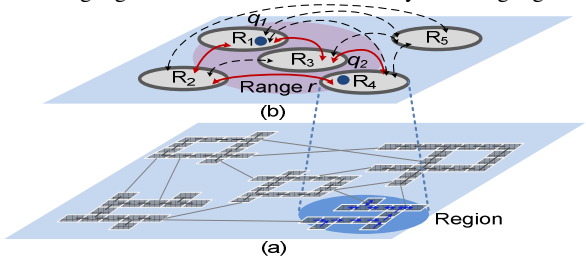


Figure 11. The scenario of the two-layer routing algorithm.

Before searching for local routes between two given cells, we first determine the region sequences to reduce searching space. By utilizing a lower bound of transition times between any two regions, we can generate region sequences with respect to two given cells. According to each region sequence, we search for possible local routes that sequentially traverse these regions. Note that the proposed A*-like algorithm is used for searching for routes between any two regions here. In Figure 11, for instance, given a location sequence $q_1 \rightarrow q_2$, and a corresponding range r, the location q_1 and the location q_2 locate in region R_1 and region R_2, respectively. There are multiple possible region combinations for searching for routes between the two locations. Although the searching space of route generation is restricted to the range r (e.g., the red part), the routes from q_1 to q_2 would possibly

traverse the regions in different orders. In Figure 11, there are many possible region sequences from q_1 to q_2 (e.g., $R_1 \rightarrow R_4$, $R_1 \rightarrow R_2 \rightarrow R_4$, $R_1 \rightarrow R_3 \rightarrow R_4$, $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4$ etc.). With utilizing lower bounds of transition times between regions, the possible region sequences would be reduced to satisfy the given time span. For instance, the qualified region sequences are marked by red edges in Figure 11 (i.e., $R_1 \rightarrow R_2 \rightarrow R_4$, $R_1 \rightarrow R_3 \rightarrow R_4$). After deriving region sequences, we search for possible routes which traverse each region sequence.

After route generation, the top-k routes are inferred, and we further refine each route using historical data points. Route refinement has three steps: data point selection, segment formulation, and segment concatenation. First, given an inferred rough route represented by a sequence of cells, we select the historical uncertain trajectories that traverse the cells in the same order as the route. Next, we extract the data points that locate in cells of the rough route from these selected uncertain trajectories, and thus derive a set of points for each cell of the route. To formulate a specific route from selected points, we adopt linear regression for the set of points of each cell to derive a segment. We then concatenate the segments in the same order as an original inferred route. Figure 10(c) shows an example of a refined route.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed RICK using real datasets, including check-in records from Foursquare and taxi trajectories. The datasets and experimental setting is presented in Section 5.1. In the experiments, we first demonstrate the results using check-in records in Manhattan. To evaluate the effectiveness of our proposed RICK, we use the dataset of taxi trajectories. In the experiments of the performance study, we compare our proposed RICK with the existing method in terms of effectiveness and efficiency. Furthermore, the experiments demonstrate the improvements of routable graph construction and route inference.

5.1 Datasets and Settings

5.1.1 Real Datasets

In this paper, we use two real datasets to conduct the extensive experiments. One is the check-in dataset from Foursquare. We collected check-in records in Manhattan, and for each user, a series of check-in records recorded in one day is regarded as a trajectory. We pruned the trajectories that contained less than three check-in records. There are totally 6,600 trajectories. The other real dataset contained 15,000 taxi trajectories in Beijing. The average sampling rate of the raw trajectories is less than one minute. To simulate uncertain trajectories, we resampled each raw trajectory such that the time interval between two consecutive resampled points of the trajectory at least exceeded a given sampling rate S. In the experiments, the sampling rate S is set from one minute to five minutes and the default S is five minutes. For example, given S=5, the time interval between two consecutive resampled points is at least five minutes or even more.

5.1.2 Metrics

To evaluate the effectiveness of our RICK, we introduce an approach to generate the ground-truth from the raw trajectories to evaluate the effectiveness of the inferred routes. For each query, the raw trajectories that satisfy the query are selected and ranked. To rank these trajectories, a raw trajectory is transformed into a sequence of road segments and the frequency of a road segment is defined as the number of distinct trajectories that traverse it. The score of a transformed trajectory $tra: r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$ is defined by $(\sum_{i=1}^{n} \mathrm{d}(r_i))/tra.length$, where r_i is a road segment and $\mathrm{d}(r_i)$ is the frequency of the road segment r_i. Hence, the selected trajectories can be ranked by their scores.

To evaluate the difference between an inferred trajectory and a raw trajectory of the ground-truth, we first apply the length-normalized dynamic time warping distance (NDTW). Given an inferred route p and a raw trajectory tra, we define the NDTW between two trajectories as $\mathrm{NDTW}(p, tra) = \mathrm{DTW}(p, tra)/p.length$ for an optimal alignment path. To further reflect the quality of inferred routes, we utilize a maximum distance (MD) between an inferred route and a raw trajectory of the ground-truth according to the discovered NDTW. MD is defined as the maximum value of the distances measured by the optimal alignment path. Therefore, the two measurements for evaluating the inferred top-k routes are defined as follows:

$\mathrm{NDTW}(\mathrm{T}, \mathrm{T}') = \mathrm{Avg}_{p_k \in \mathrm{T}} \min_{tra \in \mathrm{T}'} \mathrm{NDTW}(p_k, tra)$, and

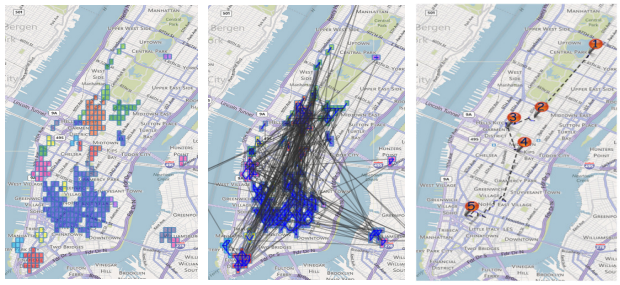
$\mathrm{MD}(\mathrm{T}, \mathrm{T}') = \mathrm{Avg}_{p_k \in \mathrm{T}} \mathrm{MD}(p_k, tra')$,

where T is the set of inferred top-k routes, T′ is the set of top-k raw trajectories, and tra'=Arg $\min_{tra \in \mathrm{T}'} \mathrm{NDTW}(p_k, tra)$.

In the experiments, the default rank threshold k is 3.

5.2 Visualization of Results

In this subsection, we use the check-in dataset in Manhattan to visualize the results derived by RICK. We first demonstrate the constructed routable graph in Figure 12 with given cell length l=500 (meters), temporal constraint θ =0.2 and minimum connection support C=3. In Figure 12, the regions are represented by different colors in Figure 12(a), and Figure 12(b) shows the edges between cells. Note that the edges within a region are drawn by blue lines, and the edges between regions are drawn by black lines. Based on the routable graph, we perform one query and let the span time be one hour for each query. Given a query as "Central Park → The Museum of Modern Art → Times Square → Empire State Building → SoHo", the top-1 route inferred by RICK is depicted in Figure 12(c). As shown in Figure 12(c), the route does not simply connect the query locations, but passes through other attractions. For example, for the partial route from "The Museum of Modern Art" to "Times Square", RICK constructs this partial route to pass by the "Rockefeller Center" based on users' historical check-in records.



(a) Regions (b) Routable graph (c) Top-1 route

Figure 12. Visualization of results in Manhattan.

5.3 Performance Study

In this section, we evaluate the performance of RICK by taxi trajectories. First, to analyze the effect of queries, the length of query location sequence $|q|$ is set from 2 to 4. In addition, a query location sequence is generated by considering a given distance between any two consecutive query locations, denoted as Δd. For a query, Δt is determined according to Δd. In the experiments, Δd is varied from 1 to 5 (in kilometers), and the corresponding Δt is set from 4 to 20 (in minutes). For each experiment, we perform almost 100 queries and averaged the results.

5.3.1 *Evaluation of Route Inference*

We compare our framework with a baseline and analyze the effectiveness of our inferred routes in different aspects.

Baseline: To evaluate the effectiveness of the discovered routes, we compare the proposed RICK with the existing approach (MPR) in [3]. In [3], given two locations (i.e., $|q|$=2), the most popular rout, which connects the two query locations, is derived. In the experiments, the parameters of MPR are set as α=2, β=2, the coherence threshold τ=0.8, and the cluster size threshold φ=20. For RICK, the settings are l=300 (meter) and k=1. Figure 13 shows the experimental results of MPR and RICK under the Taxi dataset with S and Δd varied. As shown in Figure 13(a), the error of MPR increases as S or Δd increases. It is worth mentioning that the error of RICK slightly increases as S or Δd increases, showing that RICK is able to derive the routes from uncertain trajectories. Figure 13(a) shows that RICK is more effective than MPR, although Figure 13(b) demonstrates that the query time of RICK is slightly higher than the query time of MPR.

Effect on route refinement: In the route inference of the proposed RICK, the top-k routes are derived after route generation and are further refined by route refinement. In this subsection, we compare the effectiveness of the route inference without route refinement (w/o RR) and that of the route inference with route refinement. We set k=1 and $|q|$=2 in the experiments. Figure 14 shows the error of top-1 routes by NDTW and MD. As shown in Figure 14, the errors of inferred routes increase as Δd increases. In addition, a larger l increases the error of routes discovered without route refinement. In Figure 14, with route refinement, the error of the inferred routes is obviously reduced as l increases.

Impact of data sparseness: To study the effect of the data sparseness, we calculate the number of GPS points per km^2 and derive different data sparseness by setting different S. The number of GPS points per km^2 is increased from 77 to 275 while S is decreased from five minutes to one minute. Figure 15 shows that the errors (both NDTW and MD) slightly decrease as the data sparseness increases. When the data sparseness is 275 GPS points per km^2, the errors of the inferred routes of at least 4 km (i.e., $|q|$=2 and Δd=4) are less than 500 meters and the errors of the inferred routes of at least 12 km (i.e., $|q|$=4 and Δd=4) are less than 800 meters. However, NDTW is less than 300 meters even though the data sparseness is 77 GPS points per km^2. The proposed framework is effective for inferring the top-k routes.

Efficiency: We investigated the query time of RICK and show the results in Figure 16. In the experiments, l=300 (meters), $\theta = 0.1$, C=8, S=5 (minutes), and k=3. In the route inference, we improve the efficiency of the route generation by a two-layer routing algorithm. To demonstrate the effectiveness of the two-layer routing algorithm, we compare the query time of RICK and the query time of RICK without using the two-layer routing algorithm (denoted by RICK-) in Figure 16(a) with varied $|q|$ and Δd. As shown in Figure 16(a), RICK outperforms RICK-, and the query time is obviously reduced while $|q|$ or Δd is larger. In Figure 16(b), the query time of RICK gradually increases as $|q|$ or Δd increases. However, the query time is less than one second.

5.3.2 *Evaluation of Routable Graph*

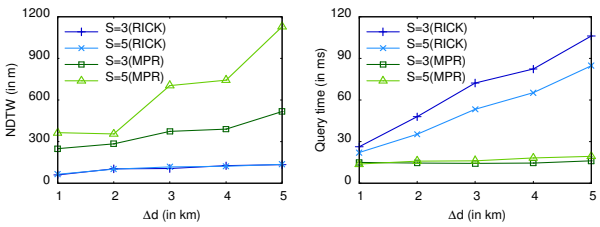
In the routable graph construction of RICK, we construct the regions referring to the connected areas and further infer and refine the moving directions within the regions. To investigate the impact of exploring shortest path on refining the routable graph, we evaluate the graph built without refinement (denoted as RG), and the graph refined by shortest path based edge inference (denoted as RG+).

To evaluate the correctness of the connectivity in a routable graph, given a raw trajectory dataset D and a graph $G =< V, E >$, the precision of connectivity in G is measured as follows:

$$|\{e|e \text{ is traversed by some } tra \in D \text{ and } e \in E\}|/|E|.$$

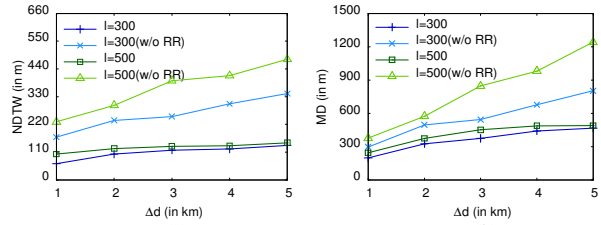
The temporal constraint θ and the minimum connection support C are used for constructing a routable graph. Hence, we analyze the precision of connectivity in the graph with varying θ and C.

In Figure 17, S is set as 5 minutes in the experiments. In Figure 17(a), C is varied from 8 to 12 and $\theta = 0.1$. Figure 17(a) shows that the precision of RG and the precision of RG+ increase as C increases. This is because a stricter constraint induces a higher



(a) (b)

Figure 13. Performance comparison of RICK and MPR.



(a) (b)

Figure 14. Effect on route refinement.

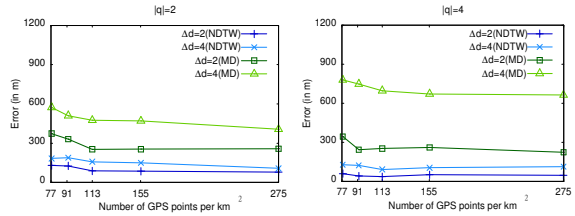
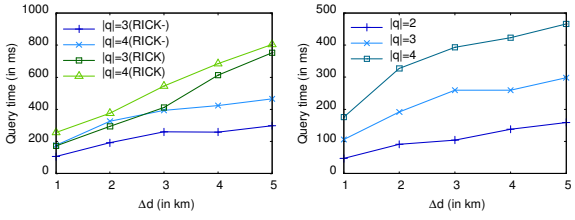
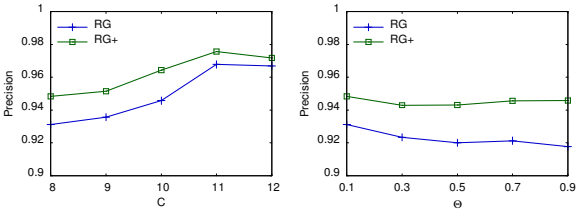


Figure 15. Effectiveness evaluation with data sparseness varied.



(a) (b)

Figure 16. Efficiency evaluation.



(a) (b)

Figure 17. Connectivity evaluation.

precision (i.e., a higher C). In Figure 17(b), C=8 and θ is varied from 0.1 to 0.9. As shown in Figure 17(b), the precision of RG and the precision of RG+ decrease as θ increases. The reason is that the precision is reduced as the constraint is loosened (i.e., a higher θ). Figure 17 depicts that the precision of RG+ is higher than that of RG, and it demonstrates that the shortest path based edge inference improves the correctness of the explored connectivity in a geographic space.

6. RELATED WORK

Route planning based on GPS trajectories: Route planning is widely investigated in [14,13,4,3] with GPS trajectories. The work [14] mainly inferred fastest routes from historical trajectories. In [13], the authors study travel route planning based on searching GPS trajectories. In [4], given a set/a sequence of locations, the top-k trajectories that best connect the given locations are retrieved from existing GPS trajectories. In [3], the authors investigated the problem of popular route planning without road network information. They introduced a transfer network model by exploiting intersections from historical GPS trajectories, and inferred the most popular route between two given locations by the turning probability of each intersection. However, these works were carried out using high sampled GPS trajectories. Given uncertain trajectories, the results obtained by [13,4] are historical uncertain trajectories and these uncertain trajectories still reveal rough routes. In addition, the trajectories derived by [4] may be far away from the query locations because these trajectories are low sampled. Using a dataset of uncertain trajectories, the accuracy of a transfer network model in [3] would be destroyed and then the effectiveness of inferred routes would be decreased.

Trip Planning based on geo-tagged social media: In recent years, mobile social applications have become popular, generating a huge volume of social media data, such as check-in records or geo-tagged photos. Such social media data can be regarded as sequences of visited locations, thereby revealing users' travel experience in terms of travel routes that link points-of-interest (POIs). Using geo-tagged photos, several studies [1,7,15] have investigated the problem of trip planning. However, the recommended trips are represented by a sequence of POIs, and the detailed route between two consecutive POIs is not specified. Different from these works, our method aims to construct the detailed route that is most likely to be taken by people by learning from the uncertain POI sequences in a mutual reinforcement way (e.g., Figure 2).

Uncertain trajectories: The research topics of trajectory uncertainty are studied in [6,10,11,16]. The work [10] introduces the problem of uncertain trajectory clustering, and focuses on the trajectory uncertainty caused by measurement errors. To reduce the uncertainty of an uncertain trajectory, the work [11] formulates an uncertain trajectory in a free space by a given maximum moving speed. However, the indistinct parts of an uncertain trajectory are enclosed in a spatio-temporal range without pointing out specific routes. In addition, the study [6] applies the techniques developed in a free space to model an uncertain trajectory in a road network. The possible routes between two sampled locations of an uncertain trajectory are restricted in a set of road segments by road network information and speed limits. Although the work [16] investigated the problem of discovering the top-k possible routes sequentially passing the queried locations from uncertain trajectories, they use road network information to reduce the uncertainty of low sampled trajectories. These works cannot derive routes from uncertain trajectories without road network information.

7. CONCLUSIONS

In this paper, we proposed RICK to infer the top-k routes traversing a given location sequence within a specified travel time from uncertain trajectories. The proposed RICK consists of the routable graph construction and the route inference. We have evaluated the proposed RICK in terms of both effectiveness and efficiency using two real datasets, check-in datasets and driving trajectories. The experiments show three aspects: 1) the inferred routes not only connect user-specified locations but also indicate detailed routes; 2) the proposed routable graph provides a good model of the uncertain trajectory dataset with an accuracy of 0.9; 3) on average, our routing algorithm can find the top-3 routes within 0.5 seconds, with a distance error smaller than 300 meters compared to its corresponding ground-truth. Meanwhile, RICK clearly outperforms the baseline by generating routes 300-700 meters closer (than those of the baseline) to the ground-truth. The experiments demonstrate the effectiveness and the efficiency of RICK. In the future, we will plan routes considering different start times and different user preferences. In addition, we will evaluate RICK by given other uncertain trajectory datasets, e.g., geo-tagged photo trips.

8. REFERENCES

[1] S. Basu Roy, S. Amer-Yahia, G. Das, and C. Yu. Interactive Itinerary Planning. In *IEEE ICDE*, pages 15-26, 2011.

[2] X. Cao, G. Cong, and C. S. Jensen. Mining Significant Semantic Locations from GPS Data. In *VLDB*, 3(1): 1009-1020, 2010.

[3] Z. Chen, H. T. Shen, and X. Zhou. Discovering Popular Routes from Trajectories. In *IEEE ICDE*, pages 900-911, 2011.

[4] Z. Chen, H. T. Shen, H. T., X. Zhou, Y. Zheng, and X. Xie. Searching Trajectories by Locations: An Efficiency Study. In *ACM SIGMOD*, pages 255-266, 2010.

[5] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Trajectory Pattern Mining. In *ACM SIGKDD*, pages 330-339, 2007.

[6] B. Kuijpers, B. Moelans, W. Othman, and A. Vaisman. Analyzing Trajectories Using Uncertainty and Background Information. In *SSDT*, pages 135-152, 2009.

[7] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *ACM CIKM*, pages 579-588, 2010.

[8] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized Travel Package Recommendation. In *IEEE ICDM*, pages 407-416, 2011.

[9] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining Periodic Behaviors for Moving Objects. In *ACM SIGKDD*, pages 1099-1108, 2010.

[10] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering Trajectories of Moving Objects in an Uncertain World. In *IEEE ICDM*, pages 417-427, 2009.

[11] R. Praing and M. Schneider. Modeling Historical and Future Movements of Spatio-temporal Objects in Moving Objects Databases. In *ACM CIKM*, pages 183-192, 2007.

[12] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *ACM SIGMOD*, pages 71-79, 1995.

[13] L.-Y. Wei, W.-C. Peng, B.-C. Chen, and T.-W. Lin. PATS: A Framework of Pattern-Aware Trajectory Search. In *UMMM*, pages 372-377, 2010.

[14] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving Directions Based on Taxi Trajectories. In *ACM SIGSPATIAL GIS*, pages 99-108, 2010.

[15] Z. Yin, L. Cao, J. Han, J. Luo, and T. S. Huang. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In *SDM*, pages 980-991, 2011.

[16] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing Uncertainty of Low-Sampling-Rate trajectories. In *IEEE ICDE*, 2012.

[17] Y. Zheng, L. Zhang, X. Xie, and W.-Y Ma, Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *WWW*, pages 791-800, 2009.

[18] Y. Zheng and X. Zhou. *Computing with Spatial Trajectories*. Springer-Verlag New York Inc, 2011.