

8-2016

# Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics

Hao WU

*Fudan University*

Jiangyun MAO

*Fudan University*

Weiwei SUN

*Fudan University*

Baihua ZHENG

*Singapore Management University, bhzheng@smu.edu.sg*


Hanyuan ZHANG

*Fudan University*

*See next page for additional authors*

**DOI:** <https://doi.org/10.1145/2939672.2939843>

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

---

## Citation

WU, Hao; MAO, Jiangyun; SUN, Weiwei; ZHENG, Baihua; ZHANG, Hanyuan; CHEN, Ziyang; and WANG, Wei. Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics. (2016). *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: San Francisco, California, August 13-17, 2016*. 1915-1924. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/3319](https://ink.library.smu.edu.sg/sis_research/3319)

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

---

**Author**

Hao WU, Jiangyun MAO, Weiwei SUN, Baihua ZHENG, Hanyuan ZHANG, Ziyang CHEN, and Wei WANG

# Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics

Hao Wu<sup>†</sup> Jianguyun Mao<sup>†</sup> Weiwei Sun<sup>†</sup> Baihua Zheng<sup>‡</sup> Hanyuan Zhang<sup>†</sup>  
Ziyang Chen<sup>†</sup> Wei Wang<sup>†</sup>

<sup>†</sup>School of Computer Science, Fudan University, Shanghai, China

<sup>†</sup>Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China

<sup>‡</sup>Singapore Management University, Singapore

{wuhao5688, jymao14, wwsun, zhanghy12, ziyangchen13, weiwang1}@fudan.edu.cn,  
bhzheng@smu.edu.sg

## ABSTRACT

Vehicle trajectories are one of the most important data in location-based services. The quality of trajectories directly affects the services. However, in the real applications, trajectory data are not always sampled densely. In this paper, we study the problem of recovering the entire route between two distant consecutive locations in a trajectory. Most existing works solve the problem without using those informative historical data or solve it in an empirical way. We claim that a data-driven and probabilistic approach is actually more suitable as long as data sparsity can be well handled. We propose a novel route recovery system in a fully probabilistic way which incorporates both temporal and spatial dynamics and addresses all the data sparsity problem introduced by the probabilistic method. It outperforms the existing works with a high accuracy (over 80%) and shows a strong robustness even when the length of routes to be recovered is very long (about 30 road segments) or the data is very sparse.

## Keywords

Trajectory; route recovery; spatio-temporal; location-based services

## 1. INTRODUCTION

With the development of GPS devices, more and more trajectory data are generated every day which brings the bloom of Location-Based Services (LBSs) [21]. To improve the quality of service, most, if not all, applications prefer a large volume of data with *zero* uncertainty. However, most of the data in the real world have uncertainty. Consequently, recovering the routes of uncertain trajectory data can help enhance the utility of data and reduce the uncertainty which can improve the performance and service quality of those trajectory-data-driven applications.

Route recovery is an important building block for many real-life applications, and we list two scenarios where route recovery could make an impact. i) According to a statistical analysis on the GPS data collected from 10,000+ taxis by [19], more than 60%

taxi trajectories are in low sampling rate (e.g., a GPS point every 2+ minutes) and reducing the uncertainty of those low sampling rate trajectories is an urgent issue. ii) Digital cameras and sensors installed in the roads are able to capture certain information of vehicles with high accuracy but those devices are still not universal. Route recovery can help to recover the trajectories passing by roads without devices installed.

In the literature, there are several works related to route recovery and they can be categorized into two groups, *non-data-driven approaches* without relying on historical data [5, 9, 12, 13, 19], and *data-driven approaches* that are based on historical data [3, 16, 20]. Non-data-driven approaches recover the routes according to geometric properties of digital map. They utilize certain external properties of road networks (e.g., the turning count, length and number of lanes) as the cost and return the optimal route having the minimum cost. These approaches model the cost empirically without any guarantee on the effectiveness as they have no historical data to infer from. Data-driven approaches recover the routes by leveraging historical data. A typical approach of finding the most popular route is introduced in [3], which is applicable to the route recovery problem, under the assumption that people tend to use the route that most people prefer. [16] tries to calibrate trajectories to some anchor points. If we use turning points as the anchor points, we can also apply the complementary component of this approach to solve route recovery problem. Another example is presented in [20], which finds the candidate routes through dynamic programming based on route popularity. To the best of our knowledge, this approach, as the state-of-the-art solution directly designed for route recovery problem, makes a comprehensive usage of historical data and outperforms many existing approaches.

We also adopt a data-driven approach to tackling route recovery problem, because data-driven approaches can draw more informative inference than geometric-based approaches, as stated in [20]. In addition, we propose to solve the problem from a probabilistic view instead of an empirical view. Empirical approaches solve the problem by intuition while probabilistic approaches can *guarantee* the effectiveness of solutions via sound theoretical models.

However, building a probabilistic model based on historical trajectory data will definitely suffer from data sparsity. It is well-known that more than 80% of the traffic in a typical city runs on only 10% to 20% of the roads hence the trajectory data are expected to be sparse. Besides, as mentioned before, given a set of trajectories, the volume of high sampling rate data based on which a probability model will be built is often much smaller than that of low sampling rate data. Inferring the probability distribution (e.g., the transition probability between roads) is essential for ev-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

KDD '16, August 13-17, 2016, San Francisco, CA, USA

© 2016 ACM. ISBN 978-1-4503-4232-2/16/08...\$15.00

DOI: <http://dx.doi.org/10.1145/2939672.2939843>

ery probability model, which can be addressed by traditional statistical frequency-based approaches. However, when the data are sparse and the volume of data is insufficient, directly counting the road frequency will result in a poor estimation according to the theory of probability. Moreover, as it is guaranteed that some roads are less frequent with insufficient trajectories passing by, missing value problem is also expected. Thus, how to address data sparsity issue will be the main challenge that our approach needs to address. Note that we will explain different types of data sparsity that route recovery has to address in details in Section 4 when we present our solution.

We summarize the key points to the route recovery problem in the following. 1) The approach should be data-driven. 2) The approach should be fully based on probability. 3) The approach should take care of the data sparsity problem. To our best knowledge, none of the existing works fulfills these three conditions simultaneously.

Motivated by this, we propose a novel route recovery system fully based on probabilistic models. Our system incorporates both temporal dynamics and spatial dynamics according to the theoretical probabilistic derivation and addresses all the challenges introduced by the data sparsity mentioned above. We include in our system a temporal model and a spatial model as the key components. The temporal model aims to estimate the travel time of a candidate route to quantify the likelihood of the candidate route being the answer. Our spatial model estimates how reasonable a candidate route is via inverse reinforcement learning that can learn the latent cost (reward) of a road through historical data. To summarize, we make three main contributions in this paper.

**Non-Empirical Approach:** We study the problem in a probabilistic view and incorporate both spatial and temporal dynamic with the theoretical probabilistic derivation.

**Data Sparsity Solution:** We propose multiple strategies to address the data sparsity problem. Our temporal model includes a new regression model to estimate the travel time of temporal-sparse trajectories, and also proposes a static-temporal separation matrix factorization approach to deal with data sparsity; the spatial model adopts inverse reinforcement learning to solve the data sparsity problem against traditional statistical frequency-based approach.

**Large Improvement:** We conduct extensive experiments using real taxi trajectories to demonstrate both the effectiveness and the robustness of our system when facing data sparsity issue. The results show that our system largely outperforms existing approaches. It achieves a high accuracy consistently (over 80%), even when the pair of GPS locations are far away from each other (e.g., about 30 road segments apart) while other approaches can only achieve an accuracy about 40%.

## 2. RELATED WORK

Many existing works [12, 13, 22] adopt shortest path to return a route with minimum weight. [22] is based on geometric and topological information of the road network; [12] defines the cost as a heuristic cost function which is related with the delay of traffic lights and left turns; [13], as an extension of [12], uses the same cost function to build a candidate graph and finds the path with as many straight lines as possible. These approaches fail to capture the preference of drivers when choosing a route, as many drivers select roads not based on whether roads are straight or not but based on the traffic condition. Differently, our system can capture the spatial dynamics by learning from historical trajectories. Map matching algorithms for low-sampling rate trajectories are proposed in [8, 19]. Although these approaches have also used the concept of transition probability, they only use this probability as a factor in a score function but the real frameworks are still empirical-based. In

brief, they are all non-data-driven approaches, and the probabilities are empirically set.

On the other hand, some existing works including [5, 7, 16, 20] adopt model-based approaches to solve the route recovery. [5] infers the route between two GPS samples by a binary logit model utilizing hidden Markov model. However, the performance of this model deteriorates when there are more than two candidate routes. [7] uses absorbing Markov chain model to synthesize routes for low sampling trajectories. Although it is a probabilistic approach, it suffers from the problem of data sparsity, as it counts the frequency in historical data to estimate the transition probability between two states. [16] constructs the transition matrix without considering the destination information which is fatal in the route recovery problem. Besides, it needs to enumerate all the possible paths which makes the computation cost extremely high when the route to be recovered is long. [20] constructs the traverse graph from historical trajectories and finds the optimal route in the graph via dynamic programming. This approach deals with data sparsity but the whole framework is empirically designed. As an contrast, each component of our system is designed with the guarantee of probability while taking great care of data sparsity. Besides, the query process of our system is optimized to support online search.

## 3. PRELIMINARY

We first present the formal definitions of *road network*, *route* and *trajectory* as following.

**DEFINITION 1 (ROAD NETWORK).** A *road network* is modeled as a directed graph  $G(V, E)$ , where  $V$  refers to the set of vertices (i.e., crossroads) and  $E$  refers to the set of edges (i.e., road segments). We assume each edge  $r \in E$  is from a vertex  $v \in V$  to another vertex  $v' \in V$ , where  $r.s = v$  and  $r.e = v'$  represent the source and the end of the edge respectively,  $r.s \rightarrow r.e$  refers to the direction, and  $r.len$  is the length of the edge  $r$ .

**DEFINITION 2 (ROUTE).** A *route*  $\mathcal{R}$  is a list of adjacent road segments,  $r_1 \rightarrow r_2 \rightarrow \dots \rightarrow r_m$ , where each two consecutive road segments are connected in  $G$ , i.e.,  $r_i.e = r_{i+1}.s$ . We use  $\mathcal{R}[i]$  to denote the  $i$ -th road segment in  $\mathcal{R}$ .

**DEFINITION 3 (TRAJECTORY).** A *trajectory*  $Tr$  is a sequence of GPS positions with timestamps, i.e.,  $Tr = \{(p_1, t_1), (p_2, t_2), \dots, (p_n, t_n)\}$ , where  $p_i$  in the form of  $(x_i, y_i)$  captures the latitude and longitude of the  $i$ -th GPS position and  $t_i$  is the timestamp.

**DEFINITION 4 (MAP MATCHING).** *Map matching* is a procedure to match the trajectory  $Tr$  to a route  $\mathcal{R}$ .  $\mathcal{R}(p_i)$  returns the road segment to which point  $p_i$  is mapped.

When the sampling rate is low, two continuous GPS sample points  $p_1, p_2$  might be mapped to two road segments  $r_1, r_2$  that are not adjacent (i.e.,  $r_1.e \neq r_2.s$ ). In other words, the moving object must pass certain unknown road segment(s) after  $r_1$  but before  $r_2$ . Take Figure 1 as an example. Given a trajectory  $Tr = \{(p_s, t_s), (p_e, t_e)\}$ , both routes  $\mathcal{R}_1$  and  $\mathcal{R}_2$  are possible. In this paper, we study the problem of *route recovery* defined in Problem 1, which locates the route  $\mathcal{R}^*$  that has the highest possibility of being taken by a given trajectory  $Tr$ . Note that although we focus on the recovery of the route between two consecutive locations, the techniques developed can be easily extended to recover the route passed by several consecutive locations in a sparse sampled trajectory.

**PROBLEM 1 (ROUTE RECOVERY).** Given a set of (high sampling rate) historical trajectories  $\mathcal{T}$  and a road network  $G$ , the *route recovery query* takes two GPS positions with timestamps (i.e.,  $(p_s, t_s)$  and  $(p_e, t_e)$ ) as input, and tries to recover the real route  $\mathcal{R}$  from  $p_s$  to  $p_e$  that is passed by  $Tr$  as accurate as possible. Note that there is no restriction that  $p_s/p_e$  can only be the source/destination of a trajectory, any two consecutive positions in a trajectory are legal to be the input.

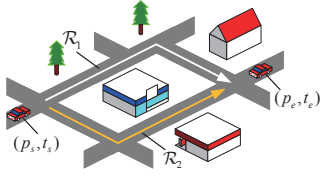


Figure 1: Example route recovery problem

## 4. SPATIO-TEMPORAL-BASED ROUTE RECOVERY SYSTEM

In this section, we introduce a novel system as our solution to perform *Route Recovery*, namely Spatio-Temporal-based Route Recovery System (STRS). STRS consists of three main components, *preprocessor*, *spatio-temporal model* and *route search engine*, as shown in Figure 2.

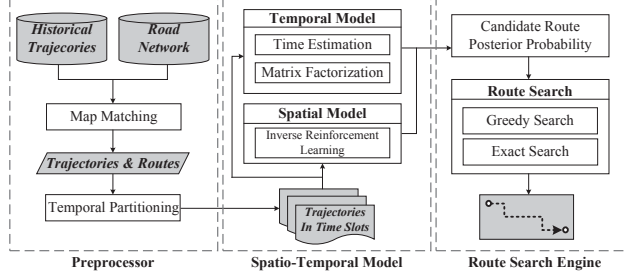


Figure 2: System architecture

**Preprocessor** performs *preprocessing* of trajectories, including map matching and trajectory temporal partitioning. **Spatio-temporal Model**, as the essential part of STRS, tries to learn and model properties of routes from historical data via a probability approach. This component contains two separate models, i.e., the *spatial model* and the *temporal model*. **Route Search Engine** computes the posterior probability of a candidate route which can be regarded as the score of a route and returns a route with the highest score.

The main objective of system STRS is to solve the route recovery problem via a pure probabilistic approach. To facilitate following discussion, we summarize the main notations used in this paper in Table 1. Given a start GPS position  $p_s$  with its timestamp  $t_s$  and an end GPS position  $p_e$  with its timestamp  $t_e$ , the route recovery problem is equivalent to find the route  $\mathcal{R}^*$  with the highest posterior probability w.r.t. the input information, i.e.,

$$\begin{aligned} \mathcal{R}^* &= \arg \max_{\mathcal{R}} P(\mathcal{R}|p_s, p_e, t_s, t_e, \mathcal{T}) \\ &= \arg \max_{\mathcal{R}} P(\mathcal{R}|p_s, p_e, t_e, \Delta t, \mathcal{T}) \end{aligned}$$

where  $\mathcal{T}$  is the set of historical trajectories and  $\Delta t = t_e - t_s$ . Based on Bayes' theorem,

$$\begin{aligned} &P(\mathcal{R}|p_s, p_e, t_e, \Delta t, \mathcal{T}) \\ &= P(\mathcal{R}, p_s, p_e, t_e, \Delta t, \mathcal{T}) / P(p_s, p_e, t_e, \Delta t, \mathcal{T}) \\ &\propto P(\mathcal{R}, p_s, p_e, t_e, \Delta t, \mathcal{T}) \\ &= P(\Delta t|\mathcal{R}, p_s, p_e, t_e, \mathcal{T}) \cdot P(\mathcal{R}|p_s, p_e, t_e, \mathcal{T}) \cdot P(p_s, p_e, t_e, \mathcal{T}) \\ &\propto P(\Delta t|\mathcal{R}, p_s, p_e, t_e, \mathcal{T}) \cdot P(\mathcal{R}|p_s, p_e, t_e, \mathcal{T}) \end{aligned} \quad (1)$$

Since  $P(p_s, p_e, t_e, \Delta t, \mathcal{T})$  and  $P(p_s, p_e, t_e, \mathcal{T})$  are the probabilities of input information, they can be regarded as constants. In the following, we omit the notation  $\mathcal{T}$  in all the representations of probabilities for presentation clarity.

Based on Equation (1), we understand that the posterior consists of two probabilities, the time interval likelihood part  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$  and the time interval-invariant posterior part  $P(\mathcal{R}|p_s, p_e, t_e)$ . The former describes how likely the time interval will be if  $\mathcal{R}$  is the route and  $t_e$  is the end time; the latter quantifies the existence probability of  $\mathcal{R}$  if the route starts from  $p_s$  and ends in  $p_e$  at time  $t_e$ .

Table 1: Major notations

Notation	Description
$r$	road segment or road element (in Section 4.2.1)
$\mathcal{R}, \mathcal{R}[i]$	route, $i$ -th road segment in route $\mathcal{R}$
$\mathcal{R}(p)$	road segment to which position $p$ is matched
$p_s/p_e$	start/end position in a route recovery query
$t_s/t_e$	start/end timestamp in a route recovery query
$\Delta t$	time interval of a route recovery query, i.e., $\Delta t = t_e - t_s$
$\mathcal{T}$	entire historical trajectory set
$\tau(t)$	time slot in which time $t$ falls
$\nu$	interval of a time slot

### 4.1 Preprocessor

The preprocessor of STRS performs two processes, *map matching* and *temporal partitioning*. We first adopt map matching algorithm to get the ground truth of historical data, i.e., the actual route. We consider all the trajectories that can be mapped to routes without uncertainty as *useful* historical samples. E.g., according to [10], when the sampling rate is 1 ~ 30s per point, map matching can achieve an accuracy about 99%, indicating that these trajectories can obtain the ground truth route without uncertainty.

It is known that the properties of historical trajectory data may vary over time [1]. This observation motivates us to partition the trajectories based on the temporal dimension. We then introduce a parameter  $\nu$  which determines the temporal duration within when the trajectories shall be gathered into one class. By default, we set  $\nu$  to 60 minutes in our experiments. Accordingly, there are 48 partitions  $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_{48}$  with  $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_{24}$  corresponding to working day and  $\mathcal{T}_{25}, \mathcal{T}_{26}, \dots, \mathcal{T}_{48}$  corresponding to weekends. For those trajectories crossing multiple time slots, we distribute them according to the time slot which  $t_e$  is in. The effects of different time slot granularity will be studied in Section 5.3.1.

### 4.2 Spatio-Temporal Model

As introduced previously, spatio-temporal model is the essential component of STRS. Recall that Equation (1) consists of two probabilities, i.e.,  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$ , and  $P(\mathcal{R}|p_s, p_e, t_e)$ . The former is to model the likelihood of the time interval that is taken care by the temporal model and the latter is the posterior of a route regardless of time information  $\Delta t$  which accounts for a spatial model to model existence of the route.

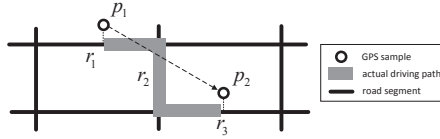
#### 4.2.1 Temporal Model

In the following, we introduce our temporal model to model the likelihood  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$ . The key is to model the distribution of the observed time interval  $\Delta t$ , conditioned by a candidate route  $\mathcal{R}$ , given  $p_s$  and  $p_e$  under time slot  $\tau(t_e)$ , where  $\tau(t_e)$  refers to the time slot that end time stamp  $t_e$  falls in. Intuitively, the distribution will be strongly correlated with the expected time of  $\mathcal{R}$  in  $\tau(t_e)$ , i.e., the closer  $\Delta t$  and the expected time of  $\mathcal{R}$  are, the higher the likelihood will be. The connection between the expected time and the distribution will be further demonstrated in Section 4.3. To be more comprehensive, we call the expected time as *estimated time* (denoted as  $\Delta t_E$ ) to emphasize that we want to perform estimation on the time cost of a candidate route  $\mathcal{R}$ , which is achieved by two steps, including *Regression Estimation Model* and *Static-dynamic-separation Matrix Factorization*, as detailed below.

**Regression Estimation Model.** This step is to estimate the time taken by a given  $\mathcal{R}$  in time slot  $\tau(t_e)$  through historical trajectories. There are several existing works on solving the travel time estimation. Most of them can be categorized into *road segment-based* and *route-based*. However, both categories face some problems in supporting route recovery. Road segment-based approaches, e.g., [4, 15], first estimate the average speed and then get the time cost of individual road segment. The travel time of a route is the summation of the time cost of road segments passed by the route. These



approaches face issues when the speed is not available or the sampling rate is not high enough. Take  $p_1$  and  $p_2$  in Figure 3 as an example. The ratio of the total travel distance between  $p_1$  and  $p_2$  to the time interval *does not* provide a good estimation of the travel speed as the route passes two junctions and it is very likely that the speed varies. However, this sample is a *useful* historical trajectory. Although the corresponding road segment of  $p_1$  and  $p_2$ , i.e.,  $r_1$  and  $r_3$ , are not adjacent, it is still easy to infer that  $r_2$  should be included in the route. On the other hand, route-based approaches, e.g., [1, 11], collect the same routes in the historical dataset and return their mean time cost as the answer. This kind of approaches can avoid estimating the speed thus they can address the problem faced by road segment-based approaches. However, they face significant data sparsity problem. Note that data sparsity is severe in route recovery as the number of historical routes that are same as  $\mathcal{R}$  presented in  $\tau(t_e)$  could be very small or even zero. In other words, although route-based approaches have better performance, they are not applicable in route recovery due to the small number of historical routes that are same as  $\mathcal{R}$  presented in  $\tau(t_e)$ . A hybrid approach that combines both road segment-based and route-based approaches is proposed in [18]. It estimates the time cost of individual driver which is over-grained and will deteriorate the data sparsity problem. Hence, we propose an approach to combine these two types of approaches by a regression model to solve the data sparsity problem.



**Figure 3: Example showing the sampling rate problem**

The main idea of our solution is that we adopt a road segment-based approach to estimate the total time cost of a route and employ a route-based approach to train the time cost of each road segment by minimizing the error of the estimation in historical data. As we intend to consider the time spent on the crossroads, notation  $r$  denotes not only road segments but also *crossroads*, which is different from its original meaning used in Definition 2. For simplicity, we name  $rs$  as *road elements*.

In detail, we first construct a matrix  $\mathcal{H} \in \mathbb{R}^{|\mathcal{T}_\tau| \times (m+n)}$ , where  $m$  and  $n$  refer to the total number of road segments and crossroads respectively.  $\mathcal{H}_{i*} \in \mathbb{R}^{m+n}$  denotes the  $i$ th row of  $\mathcal{H}$  and  $\mathcal{H}_{i*}$  represents each historical training sample. For each historical route  $\mathcal{R}^{(i)}$  processed from  $\mathcal{T}_\tau$  (for simplicity, in this section we use  $\tau$  to indicate  $\tau(t_e)$ ), we construct each training sample based on following criteria. We use superscript with parentheses, e.g.,  $(i)$ , to index the training sample. Note that  $\mathcal{R}^{(i)}[1]$  and  $\mathcal{R}^{(i)}[k^{(i)}]$  represent the first and the last road segment of  $\mathcal{R}^{(i)}$  with corresponding length  $k^{(i)} = |\mathcal{R}^{(i)}|$ .

$$\mathcal{H}_{ij} = \begin{cases} 1 & \text{if } r_j \in \mathcal{R}^{(i)} \ \& \ r_j \neq \mathcal{R}^{(i)}[1] \ \& \ r_j \neq \mathcal{R}^{(i)}[k^{(i)}] \\ \frac{\text{dist}_G(\text{proj}(p_s^{(i)}, \mathcal{R}^{(i)}[1]), \mathcal{R}^{(i)}[1, e])}{\mathcal{R}^{(i)}[1].\text{len}} & \text{if } r_j = \mathcal{R}^{(i)}[1] \\ \frac{\text{dist}_G(\mathcal{R}^{(i)}[k^{(i)}].s, \text{proj}(p_e^{(i)}, \mathcal{R}^{(i)}[k^{(i)}]))}{\mathcal{R}^{(i)}[k^{(i)}].\text{len}} & \text{if } r_j = \mathcal{R}^{(i)}[k^{(i)}] \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Here,  $\text{dist}_G(a, b)$  refers to the network distance from  $a$  to  $b$  via road network  $G$ , and  $\text{proj}(p, r)$  refers to the projection position of a point  $p$  on the road segment  $r$ . We represent a historical route  $\mathcal{R}^{(i)}$  by setting the element of  $\mathcal{H}_{i*}$  to 1 corresponding to the road segments passed by  $\mathcal{R}^{(i)}$ . Note that although a route is represented by a set of *complete* road segments, the training input representa-

tion  $\mathcal{H}_{i*}$  is different. As the start position  $p_s^{(i)}$  and the end position  $p_e^{(i)}$  may lay on the middle of the road segment, elements in  $\mathcal{H}_{i*}$  corresponding to  $\mathcal{R}^{(i)}[1]$  and  $\mathcal{R}^{(i)}[k^{(i)}]$  are set to the ratio in terms of length within the road segment passed by the trajectory. Equation (2) gives out the criteria for road segments and for crossroads we set corresponding elements in  $\mathcal{H}$  to 1 if the crossroad is passed by  $\mathcal{R}^{(i)}$ .

For all of the training data, we set the cost function as the squared error between the estimated travel time and the observed time interval as Equation (3). The first component represents the error in a matrix form, where  $\Delta T = (\Delta t^{(1)}, \Delta t^{(2)}, \dots, \Delta t^{(|\mathcal{T}_\tau|)})^\top$  and  $\phi_\tau \in \mathbb{R}^{m+n}$ . Note that  $\phi_\tau(j)$  is the estimated time cost spent on  $r_j$  in time slot  $\tau$ . Because the value of  $\phi_\tau$  may be odd in some road segments as it can be set to any value to fit the training data, we add the second component to restrict the solution so it can only vary around a given value, i.e.,  $\phi'_\tau$ .  $\phi'_\tau$  denotes a *rough* estimation of the time cost, which is estimated by the road segment-based approach. Although this estimation is not accurate, we can infer that the actual time cost of each road segment shall be not too far from this estimation. Moreover, we also exert a regularization on  $\phi_\tau$  to further avoid over-fitting.

$$C(\phi_\tau) = \frac{1}{2} \|\mathcal{H}\phi_\tau - \Delta T\|_2^2 + \frac{\lambda_1}{2} \|\phi_\tau - \phi'_\tau\|_2^2 + \frac{\lambda_2}{2} \|\phi_\tau\|_2^2 \quad (3)$$

We use stochastic gradient decent (SGD) [2] to train the model and the gradient can be computed by:

$$\nabla C = (\mathcal{H}_{i*}\phi_\tau - \Delta t^{(i)}) \mathcal{H}_{i*} + \lambda_1 (\phi_\tau - \phi'_\tau) + \lambda_2 \phi_\tau$$

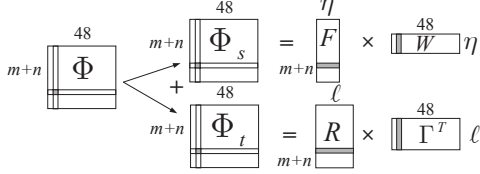
After training  $\phi_\tau$  for each individual time slot, we combine them together in column-wise, i.e.,  $\Phi = (\phi_1 \ \phi_2 \ \dots \ \phi_{48}) \in \mathbb{R}^{(m+n) \times 48}$ . Note that in order to achieve low variance, we train the road elements with the number of historical trajectories passing by *larger* than certain support count (5 in our experiment). Besides, as the dataset will further be partitioned into 48 time slots and we understand that 80% of the traffic in a typical city runs on only 10% to 20% of the roads, a *new data sparsity problem* emerges. Actually, after the training, an *incomplete* time cost matrix  $\tilde{\Phi}$  with some elements being null is generated and we will visualize the missing elements in Section 5.3.2.

**Static-dynamic-separation Matrix Factorization.** This step is to infer the value of missing elements in the cost matrix  $\tilde{\Phi}$  according to other elements. We assume the time cost of a road element consists of two costs, including a *static cost* and a *temporal-dynamic cost*, as presented in Equation (4). Static cost models the time cost that is influenced by the explicit features of the road element, such as the length of the road, the degree of a crossroad and so on. This cost is invariant over time. The temporal-dynamic cost captures the dynamic cost which varies over time.

$$\Phi = \Phi_s + \Phi_t = FW + R\Gamma^\top \quad (4)$$

where  $\Phi_s, \Phi_t \in \mathbb{R}^{(m+n) \times 48}$ .  $\Phi_s$  is the static time cost matrix and  $(\Phi_s)_{ij}$  denotes the static time cost of road element  $i$  in time slot  $j$ . As the static time cost matrix is temporal-invariant,  $(\Phi_s)_{i1} = (\Phi_s)_{i2} = \dots = (\Phi_s)_{i48}$ . We factorize it into  $FW$  where  $F \in \mathbb{R}^{(m+n) \times \eta}$  is the feature matrix of all road elements and  $\eta$  denotes the number of explicit features. The  $i^{th}$  row  $F_{i*}$  represents the features of  $r_i$ . The road segment features used in our system include i) the length of a road segment; ii) the level of the road segment (e.g., highways, parkways); and iii) the number of POIs near the road segment; while the features of crossroads used in our system are i) the degree of the crossroad; ii) the number of POIs near the crossroad; and iii) the average level of the road segments that are connected to the crossroad.  $W \in \mathbb{R}^{\eta \times 48}$  is the weight matrix which is to be trained, with stacking 48 equivalent weight vectors  $w$  in the column-wise, i.e.,  $W = (w, w, \dots, w)$ .

$\Phi_t \in \mathbb{R}^{(m+n) \times 48}$  is the temporal-dynamic cost matrix and  $(\Phi_t)_{i,j}$  denotes the temporal-dynamic cost of  $r_i$  in time slot  $j$ . We factorize  $\Phi_t$  into two low rank matrices, i.e.,  $\Phi_t = R\Gamma^\top$ . We propose a *latent road element factor matrix*  $R \in \mathbb{R}^{(m+n) \times \ell}$ , where each row of  $R$ , i.e.,  $R_{i*} \in \mathbb{R}^{\ell \times 1}$ , represents the latent factors of road segment  $r_i$  and  $\ell$  denotes the dimension of latent factors. Similarly, we introduce a *latent temporal factor matrix*  $\Gamma \in \mathbb{R}^{48 \times \ell}$  with each row  $\Gamma_{j*} \in \mathbb{R}^{\ell \times 1}$  representing the latent factors w.r.t. time slot  $j$ . Figure 4 illustrates the factorization.



**Figure 4: Illustration of matrix factorization**

By approximating the incomplete matrix  $\tilde{\Phi}$  based on Equation (4) and reconstructing the cost matrix according to learned  $W$ ,  $R$  and  $\Gamma$ , we can fill in the missing elements in  $\tilde{\Phi}$ . The cost function is the square error between the reconstructed elements and the elements.

$$J(W, R, \Gamma) = \frac{1}{2} \left\| (FW + R\Gamma^\top - \tilde{\Phi}) \circ \mathcal{M} \right\|_F^2 + \frac{\lambda_3}{2} (\|W\|_F^2 + \|R\|_F^2 + \|\Gamma\|_F^2)$$

Here,  $\mathcal{M}$  indicates not-null elements in  $\tilde{\Phi}$  and it is a 0-1 matrix satisfying that  $\mathcal{M}_{ij} = 0$  if  $\tilde{\Phi}_{ij}$  is null and 1 otherwise. The gradient is computed as follows:

$$\begin{aligned} \frac{\partial J}{\partial w} &= F_{i*} F_{i*}^\top w + (R_{i*}^\top \Gamma_{j*} - \tilde{\Phi}_{ij}) F_{i*} + \lambda_3 w \\ \frac{\partial J}{\partial R_{i*}} &= (\Gamma_{j*}^\top \Gamma_{j*} + \lambda_3) R_{i*} + (w^\top F_{i*} - \tilde{\Phi}_{ij} + \lambda_3) \Gamma_{j*} \\ \frac{\partial J}{\partial \Gamma_{j*}} &= (R_{i*}^\top R_{i*} + \lambda_3) \Gamma_{j*} + (w^\top F_{i*} - \tilde{\Phi}_{ij} + \lambda_3) R_{i*} \end{aligned}$$

We can optimize the cost function by gradient-based optimization methods [2]. After learning the weight matrix  $W$  and two latent factor matrices  $R$  and  $\Gamma$ , we can reconstruct the complete time cost matrix according to Equation (4) and estimate the travel time of a route by constructing  $H_{i*}$  according to Equation (2) and perform inner product to the  $\tau^{th}$  column of  $\Phi$ , i.e.,  $\Delta t_E = H_{i*} \Phi_{*\tau}$ . The detail of how to use the estimated time to model the probability  $P(\Delta t | \mathcal{R}, p_s, p_e, t_e)$  will be introduced in Section 4.3.

**Why data sparsity can be solved.** i) For those sparse-sampled historical trajectories, although the route segment-based estimation is inaccurate, our regression model can leverage the time information of the entire path to adjust the cost of road elements to minimize the estimation error of routes. ii) For the sparsity of identical routes, we perform the final estimation through a road segment-based view to avoid the sparsity of historical routes. iii) For the sparsity of missing values in the matrix, matrix factorization enables the inferring of the missing value based on other values, and we separate the static cost which can even work for the roads without any car passed by in any time slot.

#### 4.2.2 Spatial Model

The spatial model is to model  $P(\mathcal{R} | p_s, p_e, t_e)$  without considering the time interval  $\Delta t$ . Since this probability is only relevant to the start position  $p_s$  and the end position  $p_e$  with respect to the time slot  $\tau(t_e)$ , we can observe that it only depends on the spatial influence under the dataset of  $\mathcal{T}_{\tau(t_e)}$ . For clarity of formulas, we omit the conditional variable  $t_e$  or  $\mathcal{T}_{\tau(t_e)}$  in the following discussion. Recall that in Section 4.2.1 notation  $r$  denotes both road segments and crossroads, here we just change back the notation, i.e.,  $r$  refers to only road segments now. Suppose  $\mathcal{R}$  consists of  $k$

road segments,  $P(\mathcal{R} | p_s, p_e)$  can be equally represented as

$$P(\mathcal{R} | p_s, p_e) = P(\mathcal{R}[1] | p_s, p_e) \prod_{i=2}^k P(\mathcal{R}[i] | \mathcal{R}[i-1], p_s, p_e) \quad (5)$$

Markov assumption is often adopted on the driving behavior to facilitate the modeling [7, 10, 21]. A straightforward approach to model the transition probability is to perform the statistic.

$$P(\mathcal{R}[i] | \mathcal{R}[i-1], p_s, p_e) = \frac{N'}{N} \quad (6)$$

where  $N'$  is the count of historical trajectories starting from  $\mathcal{R}(p_s)$  and ending in  $\mathcal{R}(p_e)$  while passing  $\mathcal{R}[i-1] \rightarrow \mathcal{R}[i]$ .  $N$  is the count of trajectories starting from  $\mathcal{R}(p_s)$  and ending in  $\mathcal{R}(p_e)$  while passing  $\mathcal{R}[i-1]$ . However, this naive method suffers from a *data sparsity* problem. To be more specific, for some  $\mathcal{R}(p_e)$ , we might not be able to find many trips that pass  $\mathcal{R}[i-1]$  and are ended in  $\mathcal{R}(p_e)$ . If  $N$  is not large enough, the estimation of the transition probability can be largely affected by randomness. The effects of data sparsity when using this frequency-based approach to compute the transition probability will be illustrated in Section 5.3.3.

**Markov Decision Process and Reinforcement Learning.** In our system, we adopt a better model to address the data sparsity problem. The decision process that makes each decision based on Markov property can be modeled as a Markov Decision Process (MDP) [17]. An (deterministic) MDP is a tuple  $(S, A, \gamma, \mathfrak{R})$ , where  $S$  is *state set* of the system,  $A$  is the *action set*,  $\gamma \in [0, 1]$  is a *discount factor*, and  $\mathfrak{R}$  is the *reward function* where  $\mathfrak{R}(s)$  denotes the reward at state  $s$ . An MDP works as following. It starts at some state  $s_0$  with reward  $\mathfrak{R}(s_0)$  and performs an action and the state of system transits to  $s_1$  with collecting reward  $\mathfrak{R}(s_1)$ . It continues until it reaches the goal. Making decision at certain state is irrelevant to all of the previous states, i.e., Markov property. *Reinforcement learning* is a learning algorithm in AI to make an optimal decision with maximized expected rewards in the future [17] in an MDP. This model is similar to making decision in the crossroads when driving. We can regard each road segment  $r$  as a state and the transition  $r_i \rightarrow r_j$  between two adjacent road segments as action. The reward of each state is the negative latent cost of each road as the larger the cost is, the smaller the reward will be. Specifically, given an MDP, reinforcement learning can be performed to figure out the best policy of each state. *Optimal Value function*  $V^*(s)$  defines the maximum reward the agent can get in the future if the current state is  $s$ , i.e.,  $V^*(s) = \arg \max_{s \rightarrow s_1 \rightarrow s_2 \dots} \mathfrak{R}(s) + \gamma \mathfrak{R}(s_1) + \gamma^2 \mathfrak{R}(s_2) + \dots$

Note that the reward of each state is often set to a negative value and the reward of the destination/goal is set to 0 to avoid performing MDP infinitely through a ring. Besides,  $\gamma$  is often empirically set between 0.95 to 1.0 (0.95 in our experiments) to exert an discount on the future. *Q-function* is a function  $S \times S \mapsto \mathbb{R}$  with the definition that  $Q(\langle r_i \rightarrow r_j \rangle | \mathfrak{R}) = \mathfrak{R}(r_i) + \gamma V^*(r_j)$ . It is not hard to find that  $Q(\langle r_i \rightarrow r_j \rangle | \mathfrak{R})$  means that the reward can be received in the future if the agent makes the decision by transferring from current state  $r_i$  to another state  $r_j$ . According to [14], the transition probability from  $r_i$  to  $r_j$  can be modeled as:

$$P(\langle r_i \rightarrow r_j \rangle | \mathfrak{R}) = \frac{1}{Z_i} e^{Q(\langle r_i \rightarrow r_j \rangle | \mathfrak{R})} \quad (7)$$

The larger the future reward is if the agent decides to drive from  $r_i$  to  $r_j$ , the larger the probability to make a decision from  $r_i$  to  $r_j$  will be.  $Z_i$  is the normalization coefficient to ensure it is a probability.

As the reward  $\mathfrak{R}$  (negative latent cost) is unavailable, if we can learn  $\mathfrak{R}$  through historical routes by maximizing the likelihood or posterior using Equation (7), we can derive all the transition probabilities between any two adjacent road segments. This draws our attention to *inverse reinforcement learning (IRL)*. Therefore, we

adopt Bayesian inverse reinforcement learning (BIRL) [14] to handle this task, because BIRL has fewer hyper-parameters, easy implementation, and quick convergence. The main idea of BIRL is to compute the mean of the posterior distribution of the reward  $\mathfrak{R}$  as the answer, given observations  $\mathcal{O}$  which is the set of historical routes.

$$P(\mathfrak{R}|\mathcal{O}) \propto P(\mathcal{O}|\mathfrak{R})P(\mathfrak{R}) = P(\mathfrak{R}) \prod_{\mathcal{R}^{(i)} \in \mathcal{O}} P(\mathcal{R}^{(i)}|\mathfrak{R}) \quad (8)$$

The likelihood part  $P(\mathcal{R}^{(i)}|\mathfrak{R})$  is the simple production of Equation (7) w.r.t. each two adjacent road segments in a historical route  $\mathcal{R}^{(i)}$ . We choose uniform distribution to be the prior of  $\mathfrak{R}$ . [14] proposes *PolicyWalk* to get  $\mathfrak{R}$ , which is a Markov Chain Monte Carlo method designed for high dimension parameters. Briefly speaking, the sampler first starts from a random initial value of  $\mathfrak{R}$  and then samples a new  $\mathfrak{R}'$  drawn uniformly at random from the neighbors of current  $\mathfrak{R}$  with distance no more than the step  $\delta$ , i.e.,  $\mathfrak{R}' \sim \text{Uniform}(\mathfrak{R} - \delta, \mathfrak{R} + \delta)$ . The new  $\mathfrak{R}'$  will be accepted with probability  $\min\left\{1, \frac{P(\mathfrak{R}'|\mathcal{O})}{P(\mathfrak{R}|\mathcal{O})}\right\}$ . The mean of last few samples drawn from the Markov chain will be returned as the answer when the Markov chain converges. Please refer to [14] for the details of *PolicyWalk* and BIRL.

**Why Data Sparsity Can Be Solved:** Unlike frequency-based approaches, IRL aims to fit the whole historical routes by assigning rewards of each state (road segments). The likelihood of each observation is not only based on the reward of the next status but also considers the whole rewards towards the destination, i.e., the Q-function. Consequently, the entire preference towards the destination will be taken into consideration which is different from the edge-centric approaches that only count the ratio of transition between two consecutive edges. Thus, for those roads that are passed by a very small number of historical trajectories, IRL will assign the feasible rewards to those roads so the model can generate the historical data as likely as possible.

### 4.3 Route Search Engine

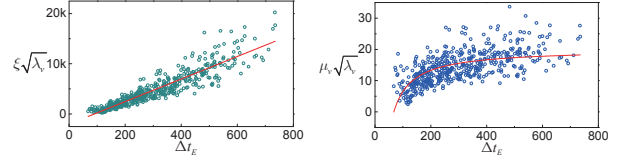
The last component accounts for locating the result of a route recovery query. It searches the routes that start from  $p_s$  and end in  $p_e$  in the road network. For each candidate route  $\mathcal{R}$ , it computes the posterior probability of  $\mathcal{R}$  according to Equation (1). The route with the highest posterior is returned as the answer. In the following, we first explain how to compute the temporal and spatial probability in Equation (1) by our novel spatio-temporal model proposed in Section 4.2, and then present how to perform the search.

**Compute  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$ .** At the first glance, one may tend to assume the distribution of the time interval of a route  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$  follows a Gaussian. However, [6] claims that the distribution of the time does not follow a Gaussian. Instead, the distribution of the speed does follow a Gaussian. i.e.,  $v \sim \mathcal{N}(\mu_v, 1/\lambda_v)$ , where  $\lambda_v = 1/\sigma_v^2$  is the precision of the Gaussian. Thus, according to [2], given the distribution of  $v$ , the distribution of  $\Delta t = \mathcal{R}.len/v$  can be derived by

$$P_{\Delta t}(\Delta t) = P_v(v) \cdot \left| \frac{dv}{d\Delta t} \right| = \frac{\xi\sqrt{\lambda_v}}{\Delta t^2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\xi\sqrt{\lambda_v}}{\Delta t} - \mu_v\sqrt{\lambda_v}\right)^2} \quad (9)$$

For better representation, we denote  $\mathcal{R}.len$  as  $\xi$ . Equation (9) implies that the distribution of  $\Delta t$  has two parameters  $\xi\sqrt{\lambda_v}$  and  $\mu_v\sqrt{\lambda_v}$  and these two parameters vary when the route is different as  $\xi$  is the length of the route and  $\mu_v$  is the average speed of the route. Note that till now, we only have obtained the expected time  $\Delta t_E$  of a route according to our temporal model. Thus, we next study the correlation between  $\Delta t_E$  and  $\xi\sqrt{\lambda_v}$  as well as  $\mu_v\sqrt{\lambda_v}$ .

Figure 5 plots the relation between  $\Delta t_E$  and the parameter  $\xi\sqrt{\lambda_v}$  as well as  $\mu_v\sqrt{\lambda_v}$  estimated from the historical routes which are



(a) Scatters of  $(\Delta t_E, \xi\sqrt{\lambda_v})$  (b) Scatters of  $(\Delta t_E, \mu_v\sqrt{\lambda_v})$

**Figure 5: Statistics for parameter estimation**

frequently passed to ensure that the estimation is accurate enough. From Figure 5(a) we can find a strong linear correlation between  $\xi\sqrt{\lambda_v}$  and  $\Delta t_E$  thus  $\xi\sqrt{\lambda_v}$  can be represented by  $a\Delta t_E + b$ . As  $\mu_v$  is the mean of the distribution of  $v$  of a certain route  $\mathcal{R}$ , we can approximate  $\mu_v$  by the division of the length of  $\mathcal{R}$  and the corresponding expected time cost  $\Delta t_E$ , i.e.,  $\mu_v \approx \xi/\Delta t_E$  and  $\mu_v\sqrt{\lambda_v} = a + b/\Delta t_E$ . Parameters  $a, b$  can be estimated by linear regression. In summary, for a candidate route  $\mathcal{R}$ , we can get  $\mu_v\sqrt{\lambda_v}$  and  $\xi\sqrt{\lambda_v}$  and further get the distribution of  $\Delta t$  by Equation (9), using the corresponding  $\Delta t_E$  computed in our temporal model and parameters  $a$  and  $b$ .

**Compute  $P(\mathcal{R}|p_s, p_e, t_e)$ .** We construct the MDP with the reward  $\mathfrak{R}_{\tau(t_e)}$  learned by our spatial model in time slot  $\tau(t_e)$ . We set the reward of destination to zero. Then we perform value iteration [17] on the MDP to get the optimal value function and Q-function of each state. According to Equation (5) and Equation (7), the probability of  $\mathcal{R}$  given  $p_s$  and  $p_e$  can be derived as

$$P(\mathcal{R}|p_s, p_e, t_e) = P(\mathcal{R}[1]|p_s, p_e) \frac{1}{Z} e^{\sum_i Q(\langle \mathcal{R}[i] \rightarrow \mathcal{R}[i+1] \rangle | \mathfrak{R}_{\tau(t_e)})}$$

**Route Search.** After explaining how to compute the posterior of a candidate route  $\mathcal{R}$ , we now discuss how to find a route as the answer, including a simple *greedy search* algorithm and an *exact search* algorithm. Recall that we have computed the Q-function of each state by value iteration before. Accordingly, the greedy algorithm starts from the start state  $\mathcal{R}[1]$  and then selects an transition action  $(\mathcal{R}[1] \rightarrow r)$  among the road segments adjacent to  $\mathcal{R}[1]$  with the highest transition probability, i.e.,  $r = \arg \max_{r'} P(\langle \mathcal{R}[1] \rightarrow r' \rangle | \mathfrak{R}) = \frac{1}{Z} e^{Q(\langle \mathcal{R}[1] \rightarrow r' \rangle | \mathfrak{R})}$ . It then transits to state  $r$ , selects the next road segment with the highest transition probability and so on. It performs state transition based on the transition probability until the destination state  $r_e$  is reached. The state sequence traversed is returned as the answer of greedy search. Note that the greedy search algorithm, in short *GreedyIRL*, is simple and fast, but the returned route might not be the optimal one as it does not consider the temporal dynamics.

Alternatively, we also propose an exact search algorithm to return the route with the highest posterior probability  $P(\mathcal{R}|p_s, p_e, \Delta t, t_e)$  based on dynamic programming. We first discretize the domain of time which is real number into integer. Assuming  $T_{max}$  is the maximum time duration of route, we construct the status matrix used in dynamic programming which is denoted as  $S \in \mathbb{R}^{m \times T_{max}}$ . The status  $S[r_i, t_j]$  refers to the log maximum route probability with constraint that the route should start from  $\mathcal{R}[1]$  and end in  $r_i$  with expected time cost  $t_j$ , i.e.,  $S[r_i, t_j] = \max_{\mathcal{R}'} \log P(\mathcal{R}'|\Omega)$ , where  $\mathcal{R}'.first = \mathcal{R}[1]$ ,  $\mathcal{R}'.last = r_i$  and  $t_j \leq \Delta t(\mathcal{R}') < t_j + 1$ . Here, for simplicity, we denote the notation  $\Omega$  to the input observations  $\{p_s, p_e, t_e\}$ . Thus, the optimal substructure can be derived as:

$$S[r_i, t_j] = \begin{cases} \log(1) & \text{if } r_i = \mathcal{R}[1] \text{ \& } t_j = \Phi(r_i) \\ \inf & \text{if } r_i = \mathcal{R}[1] \text{ \& } t_j \neq \Phi(r_i) \\ \max_{\substack{r_k \in \text{adj}(r_k) = t_j \\ \& t_i + \Phi_i \tau(t_e)}} \{S[r_k, t_i] + \log P(r_i|r_k, p_s, p_e)\} & \text{o.w.} \end{cases}$$

where  $\text{adj}(r) = \{r' | r.e = r'.s\}$ . We use a Dijkstra-like algo-



rithm to find the route with the highest posterior probability which is detailed in Algorithm 1.

Directly performing the above algorithm will suffer from great computation cost as the algorithm will not stop until all the statuses have been updated. To address this problem we maintain a lower bound  $S_{LB}$  of the result to prune the states which are impossible to have the final probability higher than  $S_{LB}$ . Notice that the probability value is always smaller than 1, which implies that the log-probability value is negative. Thus, for the status popped from the priority queue, the final log-probability of this status is definitely smaller than current status value as it will be added several log-probabilities that decrease the value. According to this property, if the value of the top status in the priority queue is already smaller than  $S_{LB}$ , we can safely confirm that all the statuses in the priority queue are impossible to be extended as the answer, as listed in Line 6-7 in Algorithm 1. We first perform the greedy route search algorithm GreedyIRL() to return an approximate optimal route as the lower bound (Line 1) and update the lower bound when the states are extended to  $r_e$  (Lines 12-14). Note that the complexity of the algorithm without pruning is  $O(CmT_{max} \times \log(mT_{max}))$ .

---

#### Algorithm 1 Exact Route Search

---

```

1:  $\mathcal{R}_{ans} \leftarrow \text{GreedyIRL}(), S_{LB} \leftarrow \log P(\mathcal{R}_{ans}|\Omega)$ ;
2:  $S[\cdot] \leftarrow -\text{inf}, S[\mathcal{R}[1], \Phi(\mathcal{R}[1])] \leftarrow 0, \mathcal{R}[\cdot] \leftarrow \text{empty array}$ ;
3:  $\text{pq.push}(S[\mathcal{R}[1], \Phi(\mathcal{R}[1])])$ ;
4: while !pq.empty do
5:    $\text{pq.pop}(S[r_k, t_l])$ ;
6:   if  $S[r_k, t_l] < S_{LB}$  then
7:     break;
8:   for each  $r_i \in \text{adj}r_k$  do
9:      $t_j \leftarrow t_l + \Phi_{i\tau}(t_e)$ ;
10:     $S[r_i, t_j] \leftarrow \max\{S[r_i, t_j], S[r_i, t_l] + \log P(r_i|r_k, p_e)\}$ ;
11:     $\mathcal{R}[r_i, t_j] \leftarrow [r_k, t_l]$ ;
12:    if  $r_i = r_e \ \& \ S_{LB} < S[r_i, t_j] + \log P(\Delta t(\mathcal{R}[r_i, t_j])|\Omega)$ 
13:      then
14:         $S_{LB} \leftarrow S[r_i, t_j] + \log P(\Delta t(\mathcal{R}[r_i, t_j])|\Omega)$ ;
15:         $\mathcal{R}_{ans} \leftarrow \text{getRoute}(r_i, t_j)$ 
16:      else
17:         $\text{pq.push}(S[r_i, t_j])$ 
18: return  $\mathcal{R}_{ans}$ ;

```

---

## 5. EVALUATION

In order to evaluate the performance of our STRS system, we conduct a comprehensive evaluation study and report the results and our findings in this section.

**Dataset Description.** We employ the real dataset generated by taxis from Porto as the main dataset, and select the central of the city which contains 2, 412 edges and 1, 410 crossroads where historical trajectories are densely distributed. The road network data is processed from OpenStreetMap. The whole dataset contains 785, 705 trajectories. As mentioned in Section 4.1, we set  $\nu$  to 60 minutes in our experiments and there are accordingly 48 time units  $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_{48}$ . In average, there are 16, 368 trajectories in each time unit  $\mathcal{T}_i$ . The average sampling rate of the original dataset is 15s per point. We split the dataset into two equal subsets, one for training and the other for testing; while we do study the impact of the size of training set in our experimental study.

**Ground Truth.** For the ground truth of a trajectory, we use the map matching algorithm of [10] to get the route in the form of a sequence of edges. For the sampling rate of 15s, it is enough to accurately ( $\approx 99\%$ ) map a series of GPS positions to a road network. Thus, we use the result of map matching as the ground truth route.

**Test Cases.** For a complete trip generated by a taxi, we subsample it by different scales (1  $\sim$  30 segments) between two consecutive

GPS positions in the subsampled trajectory. We use consecutive points in a subsampled trajectory as the input of the query. For each scale, we randomly generate 1,000 test samples. We conduct our experiments to see how the performance changes over different scales of route recovery queries, while the scale of a query is set to the total number of road segments passed by the route. Accordingly, we partition the test data based on different scales and evaluate all the approaches under different scales.

**Evaluation Criteria.** We adopt *accuracy of route recovery* as the main performance metric. It is defined as the ratio of the length of correctly inferred road segments to the length of the ground truth route  $\mathcal{R}_G$  or the inferred route  $\mathcal{R}_I$  whichever is longer, i.e.,  $\text{accuracy} = \frac{(\mathcal{R}_G \cap \mathcal{R}_I).len}{\max\{\mathcal{R}_G.len, \mathcal{R}_I.len\}}$ . We use  $\max\{\mathcal{R}_G.len, \mathcal{R}_I.len\}$  to penalize a *long* inferred route as the longer the route, the higher the chance that it contains the correct road segments.

### 5.1 Overall Evaluation

First, we compare the performance of STRS with its competitors. To have a better demonstration of the effectiveness of STRS, we implement five approaches as competitors, including HRIS, MPR, calibration, SP, FP and GreedyIRL.

*History based Route Inference System (HRIS)* is a typical data-driven route recovery approach [20]. It first locates  $k$  candidate routes between two consecutive GPS samples and then uses dynamic programming to find out the global route by picking up one route from top- $k$  routes w.r.t. each two consecutive GPS samples. As HRIS is designed to return a series of candidate routes, we conduct the evaluation on HRIS by returning different top- $k$  candidate routes, denoted as HRIS@ $k$  with  $k$  set to 1, 5 and 10. *Most Popular Route (MPR)* [3] is a data-driven approach which returns the route between two locations by observing the traveling behavior of many previous users. *Trajectory calibration* [16] is also a data-driven approach which matches the trajectory points to the anchor points and complements the missing anchor points which are very likely to be passed by the trajectory. By selecting crossroads as the anchor points, the approach can be trivially modified to solve the route recovery problem. *Shortest Path (SP)* uses the shortest path to recover the route between two locations which is commonly adopted by many applications because of its simplicity. Similar as SP, *Fastest Path (FP)* returns the route with the minimum time cost. Last but not the least, we include greedy search algorithm (GreedyIRL), introduced in Section 4.3, as our final competitor. Note that GreedyIRL is a simplified version of STRS which does not take temporal dynamics into consideration.

We evaluate the accuracy of different algorithms under various query scales from 1 to 30 segments. It is observed from Figure 6 that with the increase of the length (scale) of the query route, the accuracy of all the approaches drops. This is because, as query scale increases, the number of the possible routes between two locations increases which makes route recovery more difficult. However, among all approaches, STRS demonstrates the most robust accuracy, especially when the scale of query route is large. To be specific, when scale is 30, STRS still achieves an accuracy over 80% while others (except GreedyIRL) have their accuracy below 50%. This justifies the fact that a full probabilistic approach that can deal with data sparsity is essential. In addition, GreedyIRL, a simplified version of STRS, also demonstrates a stable performance. As the query scale becomes larger, the performance gap between GreedyIRL and STRS shrinks. This is because when the query route is short, the relative difference of time interval of different routes is large. Hence, the temporal probability  $P(\Delta t|\mathcal{R}, p_s, p_e, t_e)$  has influence on route selection. Accordingly, the advantage of STRS over GreedyIRL by considering the temporal dynamics be-

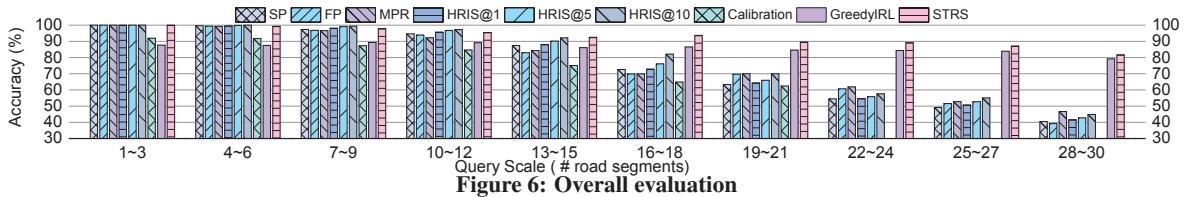


Figure 6: Overall evaluation

comes more significant. When the query route contains more segments, the relative difference of time interval of different routes is shorter. Accordingly, the temporal probability of different routes becomes more similar. In other words, the temporal influence is weakened which explains the reason that the performance gap shrinks.

Calibration performs the worst among all approaches. The main reason is that it constructs the transition matrix, i.e., the matrix denoting the transition probability from one anchor point to another one, without considering the destination. It is intuitive that the decision of turning left or right is largely depended on where the destination is. Consequently, we can utilize MDP to model the driving decision process as we also consider the destination. Besides, as calibration will enumerate all the possible routes from  $p_s$  to  $p_e$ , the computation cost scales up exponentially. Thus, when the missing route is longer than 22 road segments, we can not even get the result as the estimation of the time cost of each test sample will exceed  $10^4$  seconds.

When a query route contains less than 9 segments, all the approaches except GreedyIRL and calibration can achieve almost 100% accuracy. This is because when the query route is not too long, the ground truth of the missing route is often a direct connection of several road segments. For GreedyIRL, when the route is short, the Q-values of states for each action near the terminal do not differ much, which results in the similar transition probabilities. This explains why GreedyIRL does not achieve 100% accuracy.

HRIS recovers the route by constructing traverse graph and finding the shortest path in the traverse graph. Most of roads in the dense area are passed by historical trajectories which makes the traverse graph almost equivalent to the original road network. Thus, HRIS@1 will be reduced to SP. Our experimental results also prove this, as the accuracy difference between SP and HRIS@1 is bounded by 3%. When the route is not too long ( $<18$ ), HRIS@10 performs best among all the competitors as it leverages the information of historical data. When the length of the route becomes longer, HRIS becomes inferior to MPR. This is because HRIS is designed to recover the whole low sampling rate trajectory with GPS positions in the middle of the trajectory given; while MPR is designed to return the most popular route between two positions. Therefore, when the length of route increases, the advantage of MPR gradually emerges and finally outperforms HRIS.

MPR performs similar to FP in most of cases which indicates that popular routes are mostly fastest which is consistent with intuition. When the query route contains not too many segments (e.g.  $<18$ ), SP has a higher accuracy, as compared with FP and MPR. The reason is that when the trip is not too long, the time cost of different routes could be similar. When the query route is long, FP and MPR outperform SP since the difference of time cost of different routes becomes innegligible which affects the route selection.

## 5.2 Case Study

We next present two cases of route recovery to show how and why STRS performs better. In the first case shown in Figure 7(a), the route recovered by SP or HRIS is greatly different from that of STRS and calibration while the route recovered by STRS is the same as the ground truth. We also present the street view of some places in each route. From the photos we can figure out that the

road of the route recovered by STRS is quite wide while the road is extremely narrow in the route of SP/HRIS. STRS can capture the fact that people are reluctant to drive the route generated by SP/HRIS as it is more dangerous and unpleasant. This is because IRL can capture the reward of the road through the tendency of historical data and the reward of road segment in the dashed blue route will be assigned a relatively low value as most of drivers tend to drive the red solid route. Note that although calibration can recover the route, it takes about half an hour to return the answer.

In the second case shown in Figure 7(b), FP and MPR recover a route which seems to be reasonable and the street view also shows that the road condition of the route returned by FP/MPR is also better than STRS. However, only STRS returns a route that is the same as the ground truth. The reason is that the duration (i.e.,  $\Delta t$ ) of this query is actually much longer than the estimated time of the route recovered by FP/MPR since FP always returns the route with minimum time cost. As analyzed in Section 5.1, MPR performs similar to FP in most of time thus MPR also returns the same route as FP. From the ground truth, we intend to infer that this test sample may be generated by a driver who is unfamiliar with the road network. This case shows that the temporal model of STRS can correct the route according to the information of time duration even though the route is not efficient and natural.

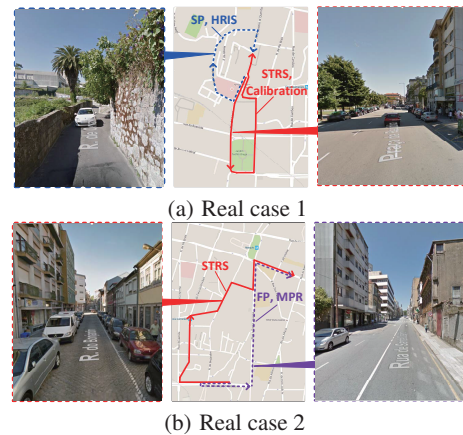


Figure 7: Case study

## 5.3 Component Study

### 5.3.1 Experiments of Preprocessor

**Experiments of Time Slot Granularity** First, we study the impact of time slot granularity in the preprocessor component. As explained in Section 4.1, parameter  $\nu$  determines the temporal duration. In our study, we set  $\nu$  to half an hour, 1 hour, 2 hours, 8 hours, 12 hours, and 24 hours. Accordingly, there are in total of  $\frac{24h}{\nu} \times 2$  time slots, half of them w.r.t. weekdays and the other half w.r.t. weekends. We also test a special case where there is only one time slot, denoted as  $\nu = 48h$ . The results are shown in Figure 8(a). We can find that with the decrease of  $\nu$ , the performance of STRS improves. As mentioned previously, information of historical data with regard to different time slots is very different. When  $\nu$  becomes larger, the time span of each time slot is enlarged and

more data are distributed into the same slot. This causes a high variance of data, which has a negative impact on STRS. When  $\nu$  is small enough, the performance starts to converge as the variance of data does not decrease any longer.

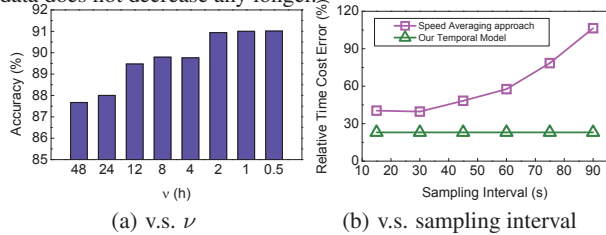


Figure 8: Results of different time spans and sampling intervals

### 5.3.2 Experiments of Temporal Model

Recall that STRS proposes a temporal model to approximate the likelihood of time duration  $\Delta t$ . In this set of experiments, we first study the effectiveness of our temporal model.

**Robustness of Time Estimation approach.** First, we conduct the experiments to show the performance of our time cost estimation approach. As route-based approaches will inevitably face the data sparsity problem in route recovery and sometimes they even cannot find any answer, we compare our approach only with road segment-based approaches and the speed of a GPS sample is estimated by the ratio of the network distance between this GPS sample and the previous one to the sampling rate. We employ the relative time cost error of a given testing route as the main metric. To be more specific, for a set of testing routes  $\mathcal{R}^{(i)}$  having time interval  $\Delta t^{(i)}$ , the relative time cost error is  $\epsilon = \sum_i \frac{|\Delta t^{(i)} - \Delta t_E^{(i)}|}{\Delta t^{(i)}}$ . Figure 8(b)

depicts the results. Our temporal model has its error rate around 20% and it consistently outperforms its competitor. This is because we adopt a regression model to adjust the cost of each road element through minimizing the error for the whole route which is *invariant* to the sampling rate. The road segment-based approach has a large error rate as the sampling interval becomes longer. Accordingly, the quality of the speed estimation drops significantly which affects the accuracy of road segment-based approach.

**Visualization of Time Cost Matrix.** Recall that to avoid high variance, STRS tries to avoid training those road segments with no or very few historical trajectories passing by. Figure 9(a) visualizes the missing value in  $\Phi$ . Elements in black are the missing values, i.e., those road segments in corresponding time slots have very few or even zero historical trajectories passing by. We can observe that data sparsity problems do exist. For example, people travel more frequently during the daytime (9:00 to 20:00) of weekdays. Accordingly, there are less missing values between 9:00 to 20:00, as compared with other time slots. In weekends, many people may stay at home which leads to the increase of missing values. Note that road segment IDs in the range of [0, 2412] represent road segments, and the IDs larger than 2,412 represent the crossroads.

After filling the missing elements in  $\Phi$  using static-dynamic separation matrix factorization, we have the entire time cost matrix  $\Phi$ . Here, we visualize the elements of  $\Phi$  via heatmap. For each row  $\Phi_i \in \mathbb{R}^{1 \times 48}$  of  $\Phi$ , which represents the time cost of road element  $i$  in different time slots, we normalize the elements of  $\Phi_i$ , i.e.,  $\Phi'_{ij} = \frac{\Phi_{ij} - \min \Phi_j}{\max \Phi_j - \min \Phi_j}$ , in order to visualize the relative trend of time cost of the same road element in different time slots, with the normalized time cost matrix  $\Phi'$  shown in Figure 9(b). The intenser the color, the smaller the time cost and vice versa. For example, we can observe that between 0:00 to 8:00 of a weekday, the color remains dark; when it reaches 9 a.m, the rush hour, the time cost increases until 20:00, which further demonstrates that our time model works and the influence of time slots can not be ignored.

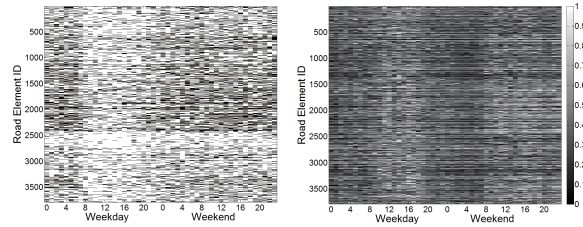


Figure 9: Visualizations of time cost matrix

### 5.3.3 Experiments of Spatial Model

**Effects of Data Sparsity.** We have pointed out in Section 4.2.2 that the key problem of spatial model is how to model the transition probability  $P(\mathcal{R}[i]|\mathcal{R}[i-1], p_s, p_e)$ . Simply using frequency-based estimation (i.e., Equation (6)) does not work, as it inevitably suffers from data sparsity problem. In the following, we conduct experiments under different degrees of data sparsity to show that our spatial model can handle the data sparsity while traditional frequency-based estimation will have problems. In detail, we use the subset of the training dataset as the sparse training dataset, with the size set to 1/2, 1/5 and 1/10 of the original training dataset.

Figure 10(a) shows when training data become sparser, the accuracy of frequency-based estimation drops drastically which justifies our analysis on data sparsity problem. On the other hand, STRS maintains a high accuracy regardless of data sparsity, which demonstrates its robustness under various quality of historical data. As explained in Section 4.2.2, performing IRL on MDP model can avoid the data sparsity problem as it is based on fitting the historical data by assigning rewards to each road and it will assign the feasible reward to those roads with no/few trajectories passed by.

**Convergence of IRL.** Next, we conduct experiments to demonstrate how an IRL model can be trained. Recall that when using PolicyWalk sampling algorithm, the step  $\delta$  is involved. We vary the parameter  $\delta$  to see how fast the Markov chain converges. We compute the posterior probability according to Equation (8) over training data to observe how IRL model fits the data. We vary the update step  $\delta$  from 0.05 to 51.2, as shown in Figure 10(b).

With the increase of the number of training iterations, the posteriors  $P(\mathfrak{R}|\mathcal{O})$  on the training data also increases w.r.t all the settings of  $\delta$  which means the model increasingly fits the training data. When  $\delta = 0.8$ , the Markov chain converges fastest and when  $\delta$  becomes smaller, the convergence slows down. This is consistent with our expectation. When  $\delta$  is very small,  $\mathfrak{R}$  is moved slightly in each iteration, which makes it slower to reach the position near the maximum posterior. When  $\delta$  becomes larger, it also increases the hardness of convergence. This is because  $\mathfrak{R}_{t+1}$  is drawn uniformly from the neighbours of previous reward  $\mathfrak{R}_t$  and most of the samples  $\mathfrak{R}_{t+1}$  drawn may be very far away from the optimal  $\mathfrak{R}$  which results in a very low acceptance ratio. Thus, many samples are rejected which directly results in the poor convergence.

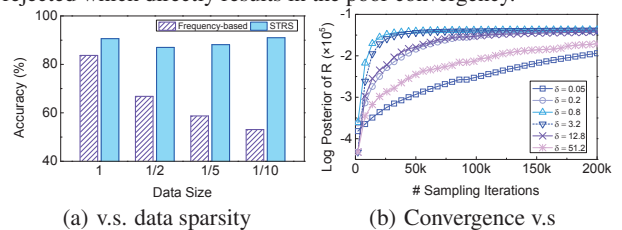


Figure 10: Evaluation on spatial model

### 5.3.4 Experiment of Route Search Engine

**Effects of Pruning Strategy** Note that we have adopted pruning strategies in the exact route search algorithm. Here, we study the



efficiency of our exact route search algorithm. To have a better illustration of the pruning strategy, we plot the query time using the exact route searching without pruning, in short  $ERS_{NoPrune}$ . Recall that the result of GreedyIRL serves as the initial value of the lower bound  $S_{LB}$ . We also plot the execution time of IRL by initializing  $S_{LB}$  to the result of shortest path. We set  $T_{max}$  to 2000 in our experiment. In Figure 11(a), both search algorithms with pruning take less time, in average about 10 times faster than  $ERS_{NoPrune}$ . The significant improvement on search time justifies the effectiveness of the pruning strategies. As  $ERS_{NoPrune}$  updates the whole status matrix  $S[:, ]$ , query scale has a less significant impact on the computation time. For  $ERS_{GreedyIRL}$  and  $ERS_{SP}$ , their search time changes w.r.t. the query scale. The reason is that when the query route contains many segments, it is very likely that the start road and the end road are far away from each other. Accordingly, the number of status that need extension or update will also increase. From the observation that  $ERS_{GreedyIRL}$  performs faster than  $ERS_{SP}$ , we can further conclude that the result of GreedyIRL does provide a better lower bound than SP and our MDP model trained by IRL is more effective.

**Efficiency Comparison** We conduct the experiment to show the time cost of online part of STRS (the route search engine) in order to see whether it can be applied in the real scenarios. The competitors are three data-driven approaches mentioned above, i.e., HRIS, MPR and calibration. As non-data-driven approaches do not require data processing and can be transferred to a shortest path algorithm, they have very small time cost (in several milliseconds) and hence are skipped in this set of experiments. Figure 11(b) depicts the result. We can see when the query scale is small, calibration is the fastest as the number of possible routes is small. However, with the query scale increases, the number of possible routes drastically increases. Thus, the cost increases exponentially, as stated in Section 5.1. MPR stays consistent with the query scale. The reason is that MPR needs a data preprocessing step for each query which costs much more than the query phase. STRS performs similar as HRIS. When scale is too large the scalability of STRS reduces for the reason that the sizes of discretized time and the states both increase. For the sake of the high accuracy STRS brings, we claim that the additional cost is worth to spend.

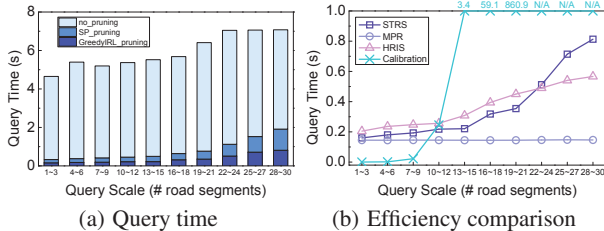


Figure 11: Evaluation on route search engine

## 6. CONCLUSION

In this paper, we study the problem of recovering the missing route of a trajectory using historical data in a probabilistic way. We propose a system based on fully probabilistic derivation showing that what kind of temporal and spatial dynamics should be taken into consideration theoretically. We have addressed all the data sparsity problems brought by the probabilistic view and thus we can take full advantages of probabilistic methods. Evaluation results show that our system outperforms all of the competitors and maintains the accuracy over 80% even when the route contains 28 ~ 30 road segments. The results also show that our system is robust for the data sparsity. In the future work, we plan to extend the route recovery to not only vehicles but also other transportations such as buses, bikes and walking to make route recovery more general.

## 7. ACKNOWLEDGEMENTS

This research is supported in part by National Natural Science Foundation of China (NSFC) under grant 61073001, Shanghai Natural Science Foundation under grant 14ZR1403100. Weiwei Sun is the corresponding author.

## 8. REFERENCES

- [1] R. K. Balan, N. X. Khoa, and L. Jiang. Real-time trip information service for a large taxi fleet. In *MobiSys 2011*.
- [2] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- [3] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *ICDE 2011*.
- [4] C. D. Fabritiis, R. Ragona, and G. Valenti. Traffic estimation and prediction based on real time floating car data. In *ITSC 2008*.
- [5] G. R. Jagadeesh and T. Srikanthan. Robust real-time route inference from sparse vehicle position data. In *ITSC 2014*.
- [6] M. Li, A. Ahmed, and A. J. Smola. Inferring movement trajectories from GPS snippets. In *WSDM 2015*.
- [7] C. Liao, J. Lu, and H. Chen. Synthesizing routes for low sampling trajectories with absorbing Markov chains. In *WAIM 2011*.
- [8] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *SIGSPATIAL GIS 2009*.
- [9] T. Miwa, D. Kiuchi, T. Yamamoto, and T. Morikawa. Development of map matching algorithm for low frequency probe data. *Transportation Research Part C*, 22, 2012.
- [10] P. Newson and J. Krumm. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL GIS 2009*.
- [11] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. Route travel time estimation using low-frequency floating car data. In *ITSC 2013*.
- [12] M. Rahmani and H. N. Koutsopoulos. Path inference of low-frequency GPS probes for urban networks. In *ITSC 2012*.
- [13] M. Rahmani and H. N. Koutsopoulos. Path inference from sparse floating car data for urban networks. *Transportation Research Part C*, 30, 2013.
- [14] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI 2007*.
- [15] J. Rice and E. V. Zwet. A simple and effective method for predicting travel times on freeways. *IEEE ITS*, 5(3), 2004.
- [16] H. Su, K. Zheng, J. Huang, H. Wang, and X. Zhou. Calibrating trajectory data for spatio-temporal similarity analysis. *The VLDB Journal*, 24(1), 2015.
- [17] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. MIT press Cambridge, 1998.
- [18] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *SIGKDD 2014*.
- [19] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive-voting based map matching algorithm. In *MDM 2010*.
- [20] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE 2012*.
- [21] Y. Zheng. Trajectory data mining: An overview. *ACM TIST*, 6(3), 2015.
- [22] Y. Zheng and M. A. Qaddus. Weight-based shortest-path aided map-matching algorithm for low-frequency positioning data. In *TRB 2011*.