Driving with Knowledge from the Physical World

Jing Yuan[1,2], Yu Zheng[2], Xing Xie[2], Guangzhong Sun[1]

[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Microsoft Research Asia, Building 2, No. 5 Danling Street, Haidian District, Beijing, P.R. China 100090
yuanjing@mail.ustc.edu.cn, {yuzheng, xingx}@microsoft.com, gzsun@ustc.edu.cn

ABSTRACT

This paper presents a Cloud-based system computing *customized* and *practically* fast driving routes for an end user using (historical and real-time) traffic conditions and driver behavior. In this system, GPS-equipped taxicabs are employed as mobile sensors constantly probing the traffic rhythm of a city and taxi drivers' intelligence in choosing driving directions in the physical world. Meanwhile, a Cloud aggregates and mines the information from these taxis and other sources from the Internet, like Web maps and weather forecast. The Cloud builds a model incorporating day of the week, time of day, weather conditions, and individual driving strategies (both of the taxi drivers and of the end user for whom the route is being computed). Using this model, our system predicts the traffic conditions of a future time (when the computed route is actually driven) and performs a self-adaptive driving direction service for a particular user. This service gradually learns a user's driving behavior from the user's GPS logs and customizes the fastest route for the user with the help of the Cloud. We evaluate our service using a real-world dataset generated by over 33,000 taxis over a period of 3 months in Beijing. As a result, our service accurately estimates the travel time of a route for a user; hence finding the fastest route customized for the user.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - *data mining, Spatial databases and GIS*;

General Terms

Algorithms, Experimentation

Keywords

Driving directions, trajectory, traffic prediction, cyber-physical.

1. INTRODUCTION

Finding fast driving routes saves both the time of a driver and energy consumption (as traffic congestion wastes a lot of gas). Meanwhile, people are more likely to choose public transportation if they can know in advance that the practically quickest driving route to a destination is still slower than the public transportation. Therefore, this service is important for both end users and governments aiming to ease traffic and protect environment. Google and Bing Maps have provided the service of finding the fastest driving path in terms of the speed constraints of a road segment. However, the practical travel time of a driving route is usually different from the result calculated based solely on speed constraints. Although real-time traffic conditions are posted on some road segments, this is meant as basic information and is not incorporated into driving direction services. It is frustrating to traverse a road segment which was fast when being checked on a map while becomes very crowded when being actually driven.

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. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08...$10.00.

Essentially, the time that a driver traverses a route depends on three aspects: 1) The physical feature of a route, such as distance, the number of traffic lights and direction turns; 2) The time-dependent traffic flow on the route; 3) A user's drive behavior. Given the same route, cautious drivers will likely drive slower than those driving aggressively. Also, users' drive behaviors vary in their progressing driving skills and experiences. E.g., traveling on an unfamiliar route, a user has to pay attention to the road signs, hence drive relatively slowly. Thus, a good routing service should consider these three aspects (routes, traffic and drivers), which are far beyond the scope of the shortest path computing.

Usually, big cities have a large number of taxicabs traversing in urban areas. To enable efficient taxi dispatch and monitoring, taxis are usually equipped with a GPS sensor, which enables them to report on their location to a server at regular intervals, e.g., 2~3 minutes. That is, a lot of GPS-equipped taxis already exist in major world cities, generating a huge volume of GPS trajectories every day [12][13][26]. Intuitively, taxi drivers are experienced in finding the quickest driving routes based on their knowledge. When selecting a route, they usually consider multiple factors including distance, traffic flows and signals, etc. Consequently, these taxi trajectories already have the knowledge of experienced drivers, physical routes and traffic conditions.

In this paper, we propose a cloud-based cyber-physical system for computing practically fast routes for a particular user, using a large number of GPS-equipped taxis and the user's GPS-enabled phone. First, GPS-equipped taxis are used as *mobile* sensors probing the traffic rhythm of a city in the *physical* world. Second, a *Cloud* in the *cyber* world is built to aggregate and mine the information from these taxis as well as other sources from the Internet like weather forecast. The mined knowledge includes the intelligence of taxi drivers in choosing driving directions and traffic patterns on road surfaces. Third, the knowledge in the *Cloud* is used in turn to serve Internet users and ordinary drivers in the *physical* world. Fourth, a *mobile* client, typically running in a user's GPS-phone, accepts a user's query, communicates with the *Cloud*, and presents the result to the user. The mobile client gradually learns a user's driving behavior from the user's driving routes and supports the *Cloud* to customize the fastest route for the user. The contribution of our work lies in three aspects:

- Using the intelligence of taxi drivers and traffic patterns mined from a large number of taxi trajectories, we propose a routing service which self-adapts to a particular user's driving behavior and customizes the fastest path for the user.
- We infer the future traffic conditions on a road using an *m*th-order Markov model considering both the historical traffic patterns and present traffic flow mined from taxi trajectories. Then, the predicted future traffic condition is integrated into the proposed routing service. We evaluated the prediction model with Beijing taxi data as well as Singapore traffic data, and found a better performance over

some well-known methods using historical patterns or real-time traffic alone like T-Drive [26] and ARIMA [10]. Using a high dimensional embedding approach, we can conduct this model online.

- We built our system with a real dataset generated by 33,000 taxis in a period of 3 months, and evaluated the system with extensive experiments in both effectiveness and efficiency.

The remainder of this paper is organized as follows. Section 2 gives an overview of our system. Section 3 presents our driving direction service. Section 4 details the processes of traffic condition prediction. Section 5 reports on major experimental results followed by some discussions. Finally, we summarize the related work in Section 6 and draw conclusions in Section 7.

2. PRELIMINARY

In this section, we define some terms used in this paper and give an overview of our work.

Definition 1 (*Taxi Trajectory*): A taxi trajectory Tr is a sequence of GPS points pertaining to one trip. Each point p consists of a longitude, latitude and a timestamp $p.t$. That is, $Tr: p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, where $0 < p_{i+1}.t - p_i.t < \Delta T$ $(1 \leq i < n)$. ΔT defines the maximum sampling interval between two consecutive points.

Definition 2. (*Road Segment*): A road segment r is a directed (one-way or bidirectional) edge that is associated with a direction symbol $(r.dir)$, two terminal points $(r.s, r.e)$, and a list of intermediate points describing the segment using a polyline. If $r.dir$ =one-way, r can only be traveled from $r.s$ to $r.e$, otherwise, people can start from both terminal points, i.e., $r.s \rightarrow r.e$ or $r.e \rightarrow r.s$. Each road segment has a length $r.length$ and a speed constraint $r.speed$, which is the maximum speed allowed on this road segment.

Definition 3. (*Route*): A Route R is a set of consecutive road segments, $R: r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$, where $r_k.end = r_{k+1}.start$, $1 \leq k < n$. The start point and end point of a route can be represented as $R.s = r_1.s$ and $R.e = r_1.e$.

Figure 1 shows the framework of our system which consists of two parts: knowledge discovery and service providing.

Knowledge discovery: This part is comprised of two steps, offline mining and online inference. 1) In the first step, we mine the accumulated historical data, including taxis trajectories and weather condition records, and build four landmark graphs respectively corresponding to different weather conditions (normal and severe weather) and day types (weekday and weekend). This mining step runs offline and not very often, e.g., every month. Here, a landmark is defined as a road segment that has been frequently traversed by taxis, and an edge connecting two landmarks represents the frequent transition of taxis between the two landmarks. Each edge in these landmark graphs is associated with a distribution of travel time learned from the taxi trajectories. Such landmark graphs can well model taxi drivers' intelligence in finding driving directions and traffic patterns on road surfaces. 2) In the online inference step, we calculate the real-time traffic on landmark edges according to the recently received taxi trajectories, and infer future traffic conditions in terms of the real-time traffic and the corresponding landmark graph. This process is conducted every 10-20 minutes.

Service providing: As shown in the left part of Figure 1, this process is comprised of five steps. 1) A user submits a query, consisting of a start point q_s, a destination q_d, a departure time t and a custom factor $\boldsymbol{\alpha}$, from a GPS-enabled mobile phone. Here, t can be a future time and $\boldsymbol{\alpha}$ is a vector, which represents how fast the user typically drives on different landmark edges. $\boldsymbol{\alpha}$ is set by a default value at the very beginning and is gradually updated in later services. 2) Using our time-dependent routing algorithm, the *Cloud* computes the fastest driving route for the user according to the received query. This routing algorithm uses the traffic condition at the time when the road was actually driven. This future condition is constantly computed in the online inference. 3) The *Cloud* sends the computed driving route along with the distributions of travel times on each landmark edge contained in the driving route to the user's mobile phone. 4) The GPS-phone records a GPS trajectory when the user really traverses the route. 5) The user's phone computes a new $\boldsymbol{\alpha}$ based on the recorded trajectory and the travel time distributions sent from the *Cloud*.

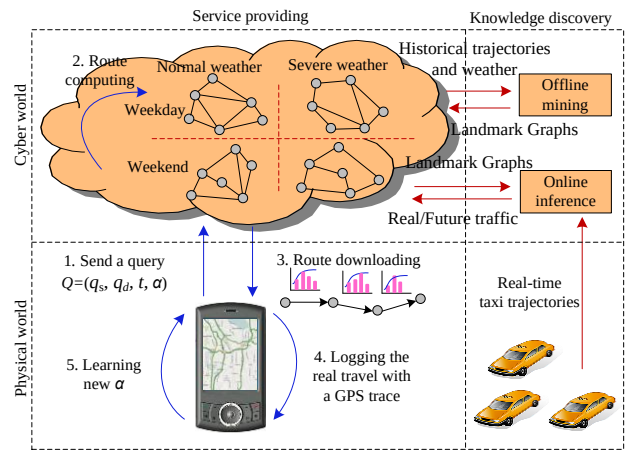


Figure 1: The framework our system

3. KNOWLEDGE DISCOVERY

3.1 Offline Mining

3.1.1 Modeling Taxi Trajectories

In practice, to save energy and communication loads, taxis usually report on their locations in a very low frequency, like 2-5 minutes per point. This increases the uncertainty of the routes traversed by a taxi [20][27]. Also, we cannot guarantee that there are sufficient taxis traversing on each road segment anytime even if we have a large number of taxis. That is, we cannot directly estimate the speed pattern of each road segment based on taxi trajectories.

In our method, we first partition the GPS log of a taxi into some taxi trajectories representing individual trips according to the taximeter's transaction records. Then, we employ our IVMM algorithm [27], which has a better performance than existing map-matching algorithms when dealing with the low-sampling-rate trajectories, to project a GPS point onto a road segment where the point was recorded. As a result, each taxi trajectory is converted to a sequence of road segments.

Based on the preprocessed taxi trajectories, we detect the top-*k* frequently traversed road segments, which are termed as *landmarks*. First, the sparseness and low-sampling-rate of the taxi trajectories do not support us to directly calculate the travel time for each road segment while we can estimate the traveling time between two landmarks (which have been frequently traversed by taxis). Second, the notion of landmarks follows the natural thinking pattern of people. For instance, the typical pattern that people introduce a route to a driver is like this "*take I-405 South at NE 4th Street, then change to I-90 at exit 11, and finally exit at Qwest Field*". Instead of giving turn-by-turn directions, people prefer to use a sequence of landmarks (like *NE 4th Street*) that highlight key directions to the destination.

After detecting the landmarks, we define the transition between two landmarks as below:

Definition 4. (*Transition*): Given a trajectory archive, a time threshold t_{max}, two landmarks u, v, arriving time t_a, leaving time t_l, we say $s = (u, v; t_a, t_l)$ is a transition if the following conditions are satisfied:
(I) There exists a trajectory $Tr: p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, after map matching, Tr is mapped to a road segment sequence $(r_1, r_2, \ldots, r_n)$. $\exists i, j,\ 1 \leq i < j \leq n$ s. t. $u = r_i$; $v = r_j$.
(II) $r_{i+1}, \ldots, r_{j-1}$ are not landmarks.
(III) $t_a = p_i.t$; $t_l = p_j.t$ and the *travel time* of this transition $t_l - t_a \leq t_{max}$.
Let S_{uv} be the set of the transitions connecting (u, v). If $S_{uv} \neq \emptyset$, we say $e = (u, v; \mathcal{T}_{uv})$ is a candidate edge, where

$$\mathcal{T}_{uv} = \{(t_a, t_l) | (u, v; t_a, t_l) \in S_{uv}\}$$

records all the historical arriving and leaving times. The frequency of e is the average number of transitions recorded per day, denoted as $e.freq$. Given a minimum frequency threshold δ, e is a landmark edge if $e.freq \geq \delta$. If no ambiguity arises, we denote the landmark edge by e_{uv}. Later, we connect all the landmark edges and construct a *landmark graph* defined as follows:

Definition 5. (*Landmark Graph*): A landmark graph $G_l = (V_l, E_l)$ is a directed graph that consists of a set of landmarks V_l (conditioned by k) and a set of landmark edges E_l conditioned by δ and t_{max}.

We observe (from the taxi trajectories) that different weekdays (e.g., Tuesday and Wednesday) almost share similar traffic patterns while the weekdays and weekends have different traffic patterns. We also find that the traffic pattern varies in weather conditions. Therefore, we respectively build different landmark graphs for weekday and weekend, and for normal and severe weather conditions, like storm, heavy rain, and snow. In total, 2×2 = 4 landmark graphs are built. The weather condition records are crawled from a weather forecast website.

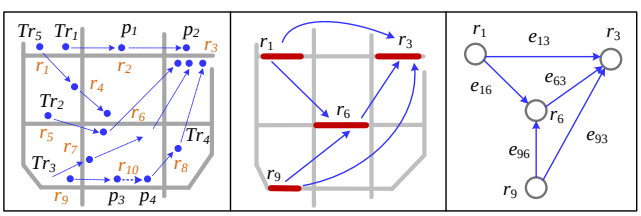


Figure 2. An example of building landmark graph

Figure 2 illustrates an example of building a landmark graph. If we set k = 4, the top-4 road segments (r_1, r_3, r_6, r_9) with more projections are detected as landmarks. Note that the consecutive points (like p_3 and p_4) from a single trajectory (Tr_4) can only be counted once for a road segment (r_{10}). This aims to handle the situation that a taxi was stuck in a traffic jam or waiting at a traffic light where multiple points may be recorded on the same road segment (though being traversed once). As shown in Figure 2 (C), after the landmark detection, we convert each taxi trajectory from a sequence of road segments to a landmark sequence, and then connect two landmarks with an landmark edge if the transitions between these two landmarks conform to Definition 5 (supposing δ =1 in this example). We propose the landmark graph to model 1) the intelligence of the experienced drivers and 2) traffic flow patterns on road surfaces during a period of historical time.

3.1.2 Mining Taxi Drivers' Knowledge

Given the transition set S_{uv} of a landmark edge e_{uv}, we aim to estimate the time-dependent travel time of e_{uv}. Figure 3 A) plots all the travel times of the transitions pertaining to a real landmark edge (on weekdays over 3 months). Clearly, the travel times gather around some values (like a set of clusters) rather than a typical Gaussian distribution. This may be induced by 1) the different number of traffic lights encountered by different drivers, 2) the different routes chosen by different drivers traveling the landmark edge, and 3) drivers' personal behavior, skills and preferences. Therefore, different from existing methods [14][22] regarding the travel time of an edge as a single-valued function based on time of day, we regard a landmark edge's travel time as a set of distributions corresponding to different time slots.

Intrinsically, different roads have different time-variant traffic patterns. That is, we cannot use a predefined time partition for all the landmark edges. Here, we employ our VE-Clustering algorithm proposed in [26] to automatically learn a proper time partition for each landmark edge based on the information entropy of the data (travel times) associated with a landmark edge. This approach consists of two phases: V-Clustering and E-Clustering. The first phase clusters the travel times pertaining to a landmark edge into several categories based on the variance of the travel times. The second phase utilizes the information gain to automatically learn a proper time partition such that in each time slot the distribution of travel time is relatively stable. As a result, the travel times of each landmark edge haven been divided into some portions (pertaining to different time slots), which are ready for distribution computing.

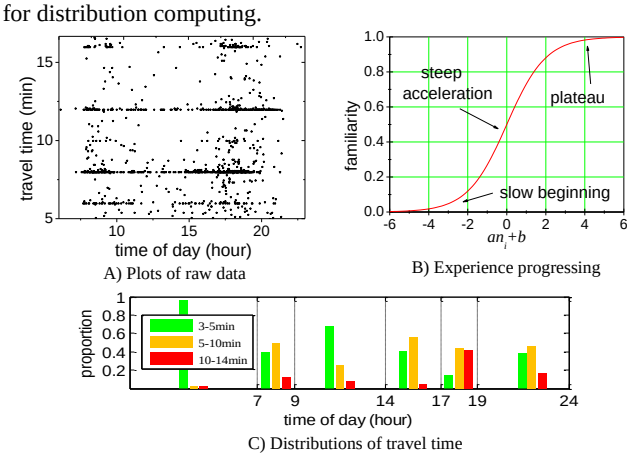


Figure 3. Learning travel time distributions from raw data

Differentiate taxi drivers' experiences: Intuitively, different taxi drivers have different knowledge in different regions of a city (especially a big city). Drivers are more likely to find out smart driving routes in a region they are very familiar with. Meanwhile, this familiarity and experience will change over the times that a driver has traveled in the region. So, when calculating the distribution of the travel times, we differentiate taxi drivers' experience based on the times they have traversed the edge.

Suppose a landmark edge e_{uv} was traversed by N different taxi drivers. Accordingly, the transition set S_{uv} can be categorized into N sample spaces. After VE-Clustering, the time of day is partitioned into several time slots. Let $\mathcal{D}_i$ be the travel time distribution (of a time slot) computed only based on the sample from a taxi driver i, denoted as

$$\mathcal{D}_i = \begin{pmatrix} 1 & 2 & \ldots & k \\ p_1^i & p_2^i & \ldots & p_k^i \end{pmatrix} \quad (1)$$

where $(1, 2, \ldots, k)$ stand for k different travel time clusters of this landmark edge and p_k^i represents the proportion of cluster k in taxi

driver i's sample space. The progress of a driver's familiarity with a landmark edge is modeled using a Sigmoid learning curve [16] (as shown in Figure 3 B), defined as:

$$f(n_i) = \frac{1}{1+e^{-(an_i+b)}}; \tag{2}$$

where $f(n_i)$ is the familiarity, a, b are the coefficients, and n_i is the times that the driver i has traversed the landmark edge. $an_i + b$ is the linear transformation which maps n_i from [*min*, *max*] to [-6, 6], where *min* and *max* respectively represent the minimum and maximum number of transitions (generated by all the drivers) on this landmark edge. Then distribution of this time slot, denoted by $\mathcal{D}$, is computed by the weighted average:

$$\mathcal{D} = \begin{pmatrix} 1 & 2 & \dots & k \\ \sum w_i p_1^i & \sum w_i p_2^i & \dots & \sum w_i p_k^i \end{pmatrix}, \tag{3}$$

Where w_i is a normalized familiarity of the driver i, calculated as

$$w_i = \frac{f(n_i)}{\sum_{i=1}^{N} f(n_i)}. \tag{4}$$

Using this method, we obtain the travel time distribution of each time slot for each landmark edge. For example, as shown in Figure 3 C), in the time slot 9-14, over 60 percent of drivers traverse the landmark edge in 3-5 minutes (the green bar), while about 30 percent of drivers need 5-10 minutes (the yellow bar) and the rest of them even spends 10-14 minutes (the red bar).

3.2 Online Inference

In this section, we infer the traffic condition at a future time (F) in terms of the landmark graphs built from historical data (H) and real-time traffic flow calculated based on recently received taxi trajectories (R). In short, $H + R \rightarrow F$. In our method, we model this problem as an mth-order Markov chain, and implement the model on-the-fly using a high-dimensional embedding approach.

Table 1 Notations

τ	Traffic updating frequency	φ	Delay (future) time
x_t	A traffic condition at time t	$\mathbf{\Omega}$	The collection of x_t
$\mathbf{X}$	Traffic condition space	$\mathbf{S}$	A finite state space
$\mathbf{F}(x)$	A discretization function $\mathbf{F}: \mathbf{X} \rightarrow \mathbf{S}$	Y_i	A random variable in $\mathbf{S}$
y_i	Realization of Y_i, $y_i = \mathbf{F}(x_{t_i}) \in \mathbf{S}$.	m	Order of a Markov chain
$\mathbf{P}^{(h)}$	h-step-ahead transition matrix of the mth-order Markov chain in $\mathbf{S}$		
$\mathbb{P}^{(h)}$	h-step-ahead transition matrix of the 1st-order Markov chain in $\mathbf{S}^m$		
$P_{a\rightarrow b}^{(h)}$	The h-step-ahead transition probability of a Markov chain from state a to state b.		

3.2.1 Modeling Traffic Condition

We track the traffic condition $x_{t_1}, x_{t_2}, \dots, x_{t_n}$ at each time stamp t_i, as shown in Figure 4. Here, x can be the average velocity that vehicles can traverse on a road segment, or the average travel time of a landmark edge. This time series of real-time traffic can be calculated based on the recently received taxi trajectories and/or road sensors, using some approaches. One method is calculating the average speed or travel time of the samples on a road.

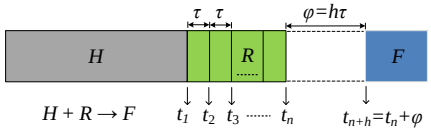


Figure 4: The framework for traffic prediction

Typically, the traffic condition is updated at a certain frequency τ,

$$\tau = t_{j+1} - t_j \qquad j = 1,2,\dots,n-1. \tag{5}$$

Given the accumulated historical traffic conditions $\mathbf{\Omega}$, we aim to predict the traffic condition at a future time $t_n + \varphi$, where

$$\varphi = h\tau, \qquad h=1, 2,\dots \tag{6}$$

In practice, the *delay* φ is configured by user-sending queries, whereas τ is often determined by a traffic monitoring system.

Since the traffic condition is usually presented to end users using discrete states, we map the continuous x_t value into a finite state space $\mathbf{S}$ by a *discretization function* $\mathbf{F}: \mathbf{X} \rightarrow \mathbf{S}$. For example, after the VE-Clustering algorithm, the travel times of transitions pertaining to a landmark edge are discretized into a cluster set, which can be regarded as the state space. After the discretization, $x_{t_1}, x_{t_2}, \dots, x_{t_n}$ is converted into a state sequence, which can be considered as the realization of a stochastic process $\{Y_i | Y_i \in \mathbf{S}\}_{i=1}^n$, where each Y_i is a random variable. Intuitively, the traffic condition at t_i usually depends on the time a short period before t_i, e.g., 1-2 hours, i.e., the past m states. Hence, we model this stochastic process as an mth-order Markov chain [24], stated as:

$$\begin{aligned} &P(Y_n = y_n | Y_1 = y_1, Y_2 = y_2, \dots, Y_{n-1} = y_{n-1}) \\ &= P(Y_n = y_n | Y_{n-m} = y_{n-m}, \dots, Y_{n-1} = y_{n-1}) \end{aligned} \tag{7}$$

$\forall y_i \in \boldsymbol{S}$, $i = 1,2,\dots,n$. Thus, for simplicity, we define our problem as: Predict the distribution of Y_{m+h} given the realization of $\{Y_i\}_{i=1}^m$, i.e., compute the h-step-ahead transition probability

$$P(Y_{m+h} = y_{m+h} | Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m), \tag{8}$$

3.2.2 High Dimensional Embedding

Our solution is comprised of the following four steps:

***Step* 1**: Compute 1-step-ahead transition matrix of the mth-order Markov chain in $\mathbf{S}$ space using the Bayesian probability model.

We denote the h-step-ahead transition probability

$$P(Y_{m+h} = y'_m | Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m),$$

by notation $P_{y_1,y_2,\dots,y_m \rightarrow y'_m}^{(h)}$. Then the h-step-ahead transition matrix of the mth-order Markov chain is denoted by

$$\mathbf{P}^{(h)} = \{P_{y_1,y_2,\dots,y_m \rightarrow y'_m}^{(h)}\}_{|\mathbf{S}|^m \times |\mathbf{S}|} \tag{9}$$

For $h = 1$, we can compute the 1-step-ahead transition matrix $\mathbf{P} = \mathbf{P}^{(1)}$ based on the Bayesian probability theory. That is,

$$\begin{aligned} P_{y_1,y_2,\dots,y_m \rightarrow y'_m}^{(1)} &= \frac{P(Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m | Y_{m+1} = y'_m) P(Y_{m+1} = y'_m)}{\sum_{j \in \mathbf{S}} P(Y_{m+1} = j)\, P(Y_1 = y_1, Y_2 = y_2, \dots, Y_m = y_m | Y_{m+1} = j)} \\ &= \frac{|\{x_{t_i} \in \mathbf{\Omega} | F(x_{t_i}) = y_1, \dots, F(x_{t_{i+m}}) = y'_m\}|}{|\{x_{t_i} \in \mathbf{\Omega} | F(x_{t_i}) = y_1, \dots, F(x_{t_{i+m-1}}) = y_m\}|}. \end{aligned} \tag{10}$$

The numerator and denominator of Equation 10 are calculated based on the statistics in the historical data when implementing.

***Example* 1**. Figure 5 shows an exemplary 1-step-ahead transition matrix $\mathbf{P} = \mathbf{P}^{(1)}$, where $m = 2$ and the state space is S={1, 2} (1 could be the normal traffic and 2 could indicate a traffic jam). For instance, the element (at row 11 and column 2) $P_{11\rightarrow 2}$= 0.205 is the transition probability from $\{Y_1 = 1; Y_2 = 1\}$ to $Y_3 = 2$. The value of $P_{11\rightarrow 2}$ is calculated according to Equation 10.

$$\begin{array}{c} \\ 11 \\ 12 \\ 21 \\ 22 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 0.795 & 0.205 \\ 0.957 & 0.043 \\ 0.523 & 0.477 \\ 0.880 & 0.120 \end{pmatrix} \end{array}$$

Figure 5: P = $\mathbf{P}^{(1)}$

$$\begin{array}{c} \\ 11 \\ 12 \\ 21 \\ 22 \end{array} \begin{array}{c} \begin{array}{cccc} 11 & 12 & 21 & 22 \end{array} \\ \begin{pmatrix} 0.795 & 0.205 & 0 & 0 \\ 0 & 0 & 0.957 & 0.043 \\ 0.523 & 0.477 & 0 & 0 \\ 0 & 0 & 0.880 & 0.120 \end{pmatrix} \end{array}$$

Figure 6: $\mathbb{P} = \mathbb{P}^{(1)}$

***Step* 2**: Embed the mth-order Markov chain into an $\mathbf{S}^m$ space by binding the consecutive m variables from $\{Y_i\}_{i=1}^n$ into a vector.

Basically, when h >1, we can still use Equation 10 to compute the h-step-ahead transition matrix. However, this process is very time consuming as we need to scan the whole sample space $\mathbf{\Omega}$ many times for different h. Instead, we compute the h-step-ahead probability by mapping the mth-order Markov chain $\{Y_j\}$ from a

space $\mathbf{S}$ into an m dimensional space $\mathbf{S}^m$ according to Lemma 1, as illustrated in Figure 7.

LEMMA 1. *Let $\vec{Y}_j = \{Y_j, Y_{j+1}, \dots, Y_{j+m-1}\}$ be a random vector of dimension m, where $\{Y_j\}_{j=0}^{\infty}$ is a mth-order Markov chain in the space $\mathbf{S}$, then $\{\vec{Y}_j\}_{j=1}^{\infty}$ is a* 1st-*order Markov chain in the space $\mathbf{S}^m$.*

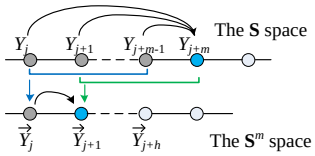


Figure 7: Mapping an *m*th-order Markov chain from S to $\mathbf{S}^m$ spaces

***Step* 3**: Compute the h-step-ahead transition matrix $\mathbb{P}^{(h)}$ of the converted Markov chain in the $\mathbf{S}^m$ space.

According to Lemma 1, $\{\vec{Y}_j\}_{j=1}^{\infty}$ is a 1st-order Markov chain in the $\mathbf{S}^m$ space. We denote the h-step-ahead transition matrix in $\mathbf{S}^m$ by:

$$\mathbb{P}^{(h)} = \{P^{(h)}_{y_1,y_2,\dots,y_m \to y'_1,y'_2,\dots y'_m}\}_{|\mathbf{S}|^m \times |\mathbf{S}|^m} \tag{11}$$

for $\forall y_j, y_j{}' \in \mathbf{S}$, j=1, 2,…, m.

Let $\mathbb{P} = \mathbb{P}^{(1)} = \{P^{(1)}_{y_1,y_2,\dots,y_m \to y'_1,y'_2,\dots y'_m}\}_{|\mathbf{S}|^m \times |\mathbf{S}|^m}$ be the 1-step-ahead transition matrix of $\vec{Y}_j$ in the embedded space $\mathbf{S}^m$, i.e.,

$$P^{(1)}_{y_1,y_2,\dots,y_m \to y'_1,y'_2,\dots y'_m} = P(\vec{Y}_{j+1} = \{y'_1, y'_2, \dots y'_m\} | \vec{Y}_j = \{y_1, y_2, \dots, y_m\})$$
$$= \begin{cases} P_{y_1,y_2,\dots,y_m \to y'_m} & \text{if } \{y_2, y_3, \dots, y_m\} = \{y'_1, y'_2, \dots y'_{m-1}\} \\ 0 & \text{else} \end{cases} \tag{12}$$

where $P_{y_1,y_2,\dots,y_m \to y'_m}$ is the element on row $y_1, y_2, \dots, y_m$ and column y'_m of the transition matrix $\mathbf{P}$.

According to the the Chapman-Kolmogorov equation [15], the h-step-ahead transition matrix for $\vec{Y}_j$ is $\mathbb{P}$ multiplied by itself h times, i.e., $\mathbb{P}^{(h)} = \mathbb{P}^h$. So, $P^{(h)}_{y_1,y_2,\dots,y_m \to y'_1,y'_2,\dots y'_m}$ is an element of $\mathbb{P}^h$ at the row $y_1, y_2, \dots, y_m$ and column $y'_1, y'_2, \dots y'_m$.
Note that this property does not hold for the original mth-order Markov chain. That's the reason why we embed the mth-order Markov chain into the $\mathbf{S}^m$ space.

***Example* 2**. Figure 6 presents the 1-step-ahead transition matrix $\mathbb{P}$, which is constructed by applying Equation (12) to the matrix $\mathbf{P}$ shown in Figure 5. For example,

$$P_{11\to12} = P(\vec{Y}_{j+1} = \{1,2\} | \vec{Y}_j = \{1,1\})$$
$$= P(Y_{j+1} = 1, Y_{j+2} = 2 | Y_j = 1, Y_{j+1} = 1) = P_{11\to2} = 0.205. \text{ Yet,}$$

$$P_{11\to21} = P(\vec{Y}_{j+1} = \{2,1\} | \vec{Y}_j = \{1,1\})$$
$$= P(Y_{j+1} = 2, Y_{j+2} = 1 | Y_j = 1, Y_{j+1} = 1) = 0,$$

Since Y_{j+1} =2 and $Y_{j+1} = 1$ never appear in the historical data $\mathbf{\Omega}$.

***Step* 4**: Compute the h-step-ahead transition matrix $\mathrm{P}^{(h)}$ in $\mathbf{S}$ space based on the $\mathbb{P}^{(h)}$.

Given $\mathbb{P}$ and $\mathbb{P}^{(h)}$, the h-step-ahead transition matrix of $\{Y_j\}$ in the $\mathbf{S}$ space can be computed directly as follows.

***Example* 3**. Figure 8 presents the matrix $\mathbb{P}^{(5)}$=$\mathbb{P}^5$, i.e., $\mathbb{P}$ to the power of 5, which is the 5-step-ahead transition matrix of $\{\vec{Y}_j\}$ in the $\mathbf{S}^2$ space (m=2). In the original $\mathbf{S}$ space, note

$$P^{(5)}_{11\to1} = P^{(5)}_{11\to11} + P^{(5)}_{11\to21} = 0.566 + 0.203 = 0.769,$$

Where $P^{(5)}_{11\to11}$ and $P^{(5)}_{11\to21}$ are obtained from $\mathbb{P}^5$. In this way, we compute other elements in $\mathbf{P}^{(5)}$, shown in Figure 9.

Formally, we have

$$\begin{array}{c|cccc} & 11 & 12 & 21 & 22 \\ \hline 11 & 0.566 & 0.221 & 0.203 & 0.010 \\ 12 & 0.517 & 0.179 & 0.290 & 0.014 \\ 21 & 0.565 & 0.247 & 0.179 & 0.009 \\ 22 & 0.520 & 0.188 & 0.279 & 0.013 \end{array}$$

Figure 8: $\mathbb{P}^{(5)} = \mathbb{P}^5$

$$\begin{array}{c|cc} & 1 & 2 \\ \hline 11 & 0.769 & 0.231 \\ 12 & 0.807 & 0.193 \\ 21 & 0.744 & 0.256 \\ 22 & 0.799 & 0.201 \end{array}$$

Figure 9: $\mathbf{P}^{(5)}$

$$P^{(h)}_{y_1,y_2,\dots,y_m \to y'_m} = \begin{cases} \sum_{y'_{m-h+1},\dots,y'_{m-1}} P^{(h)}_{y_1\dots y_m \to y_{h+1},\dots,y_m,y'_{m-h-1},\dots,y'_{m-1},y'_m} & \text{if } h < m \\ \sum_{y'_1,\dots,y'_{m-1}} P^{(h)}_{y_1\dots y_m \to y'_1,\dots,y'_{m-1},y'_m} & \text{if } h \geq m \end{cases} \tag{13}$$

In this way, we only need to pre-compute the 1-step-ahead transition matrix $\mathbf{P}^{(1)}$ of $\{Y_j\}$ in $\mathbf{S}$ space while $\mathbb{P}^{(h)}$ and $\mathbf{P}^{(h)}$ can be calculated online (h >1), which is more efficient than using a Bayesian probability model like Equation 10. The time cost for computing the matrix $\mathbf{P}^{(h)}$ is $O(h \cdot |\mathbf{S}|^{\varepsilon m})$, where ε < 2.376 [4]. In the implementation, since both m and $|\mathbf{S}|$ (number of states) are small (e.g., $m \leq 3$; $|\mathbf{S}| \leq 5$), the online computation is affordable. To further improve the efficiency, we can compute $\mathbf{P}^{(h)}$ with some small h=2, 3 in advance.

As a result, we obtain the future traffic condition (a distribution of states) φ ahead of the present time. The value (a representative travel time) of a state can be calculated in terms of the mean of the samples pertaining to the state. Though in our system this condition is represented as a distribution of travel times associated with a landmark edge at time $t + \varphi$, the method can be generally applied to other datasets and traffic prediction problems.

4. SERVICE PROVIDING

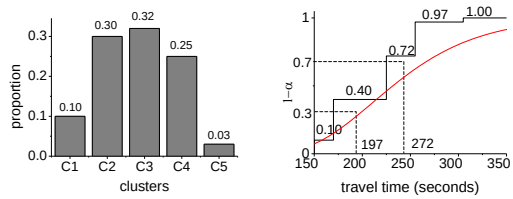
This section details the service providing process, which consists of 5 steps, as shown in the left part of Figure 1:

1) *Query Sending*. A user sends a query $(q_s, q_d, t_d, \boldsymbol{\alpha})$ to the Cloud. Specifically, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_k)$, $\forall \alpha_i \in \boldsymbol{\alpha}$, $0 < \alpha_i \leq 1$ (typically stored in a mobile phone) is a custom factor indicating how fast a user usually drives on the ith landmark edge, and k denotes the number of landmark edges. The larger the value α_i has, the faster the user drives on the ith edge. Initially, each α_i can be set as a default value, and be gradually adapted to the user's driving habits in terms of the user's driving paths collected later.

2) *Route Computing*. In this step, the Cloud first chooses a proper landmark graph according to the day type and weather of the departure time t_d. Then, a two-stage routing algorithm is performed to find out a time-dependent fastest route. In the first stage, we search the landmark graph (see Figure 2 C for an instance) for a rough route represented by a sequence of landmarks, using a time-dependent routing algorithm, like [5]. For example, Figure 10 A) depicts the travel time distribution of a landmark edge i in a given time slot, where $(c_1 \sim c_5)$ denotes 5 categories of travel times. Then, we convert this distribution into a cumulative frequency distribution function and fit it with a continuous cumulative frequency curve [3] depicted in Figure 10 B). Given a user's custom factor α_i= 0.7 of this landmark edge, we can particularly determine the travel time of the user on this landmark edge and in this time slot. Note that the traffic conditions (travel time distributions) on a landmark edge at a future time are computed using the method proposed in Section 3.2. For instance, we can respectively pre-calculate the travel time distribution of a landmark edge at the time that is 15, 30, 45, and 60 minutes later then the present time. Then, in the routing algorithm, we can choose the distribution of the time slots according to the time that the user will arrive at the landmark edge.

In the second stage, we perform a detailed routing that finds the fastest path (based on speed constraints) connecting consecutive landmarks in the rough route generated in the first stage. This

two-stage routing algorithm is even more efficient than existing methods. First, the rough routing on a landmark graph is very fast as a landmark graph is only a subset of the original road network. Second, the search space of the detailed routing becomes smaller than before as the distance between two landmarks is shorter than that between the original start and end points.



A) Travel time distribution B) Cumulative frequency distribution

Figure 10: Travel time w.r.t. custom factor

3) *Route Downloading* and 4) *Path Logging*. The Cloud sends the computed driving routes along with the travel time distributions of the landmark edges contained in the driving route to the phone. Later, the mobile phone logs the user's driving path with a GPS trajectory, which will be used to recalculate the user's custom factor $\boldsymbol{\alpha}$. The more a driver uses this system, the deeper this system understands the driver; hence, a better driving direction service can be provided.

5) *Adapting custom factor*. For simplicity, we choose one $\alpha_i \in \boldsymbol{\alpha}$ to demonstrate the updating process. Initially, we assign α_i a default value, e.g., 1.0. Let ${\alpha_i}^{(M)}$ be the α_i the client sent to the cloud at the M-th query, and $CDF_i(t)$ be the cumulative distribution function (refer to Figure 10B for an example) of the ith landmark edge. After traveling, we calculate the real travel time of the landmark edges ${T_i}^{(M)}$ by the recorded GPS traces. Then the mobile client computes the new custom factor by:

$$\tilde{\alpha}_i^{(M)} = CDF_i\left({T_i}^{(M)}\right) \tag{14}$$

To obtain a stable value for α_i, we need to study the most recent n driving routes of a user instead of a single trip. Meanwhile, the most recent driving paths should be more valuable in calculating α_i than those distant past. Therefore, we compute the new α_i by using a weighted moving average(WMA) shown as below

$${\alpha_i}^{(M+1)} = \frac{\sum_{j=1}^{n} j\tilde{\alpha}_i^{(M-n+i)}}{\sum_{j=1}^{n} j} = \frac{2}{n(n+1)}\sum_{j=1}^{n} j\tilde{\alpha}_i^{(M-n+j)} \tag{15}$$

where n is the window size of the moving average.

In the next query, the ${\alpha_i}^{(M+1)}$ will be sent to the Cloud. Note that both path recording and $\boldsymbol{\alpha}$ learning are performed in a user's mobile phone. Therefore, the user's privacy is preserved.

5. Evaluation

Considering that the travel time of a driving route depends on route, traffic and driver, we evaluated the following two aspects in the experiments. 1) Does our method precisely predict the future traffic conditions? 2) Does our method learn a user's diver behavior accurately and estimate the travel time of a route for the user precisely? If the answers are yes, our service is valuable.

5.1 Datasets

Taxi Trajectories: We build our system using GPS trajectories generated by 33,000 taxis over a period of 3 months. The total distance of the dataset is over 400 million kilometers and the total number of GPS point reaches 790 million. The average sampling interval of the dataset is 3.1 minutes and the average (Euclidian) distance between two consecutive points is about 600 meters.

Road Network: The adaptive routing is based on the road network of Beijing which has 106,579 road nodes and 141,380 segments.

Singapore traffic data: This dataset includes the updates (in a frequency of every 26 minutes on average) of traffic conditions on 50 road segments in Singapore from Nov. 1- Dec. 13 (43 days).

5.2 Evaluation on Traffic Prediction

Framework: 1) Prediction on a landmark edge: We use the taxi trajectories of the first two months as a training set (for offline mining) and choose 12 days, consisting of 6 workdays and 6 weekends, from the trajectories of the third month as a test set. 6 out of the 12 days had normal weather conditions, and the remainder had severe weather conditions. We use the expectation of the travel times as a predictor calculated based on the inferred distribution. The ground truth of a given landmark edge is computed in terms of the average travel time of the transitions (from the test dataset) pertaining to the landmark edge in the time slot to be inferred. 1,000 landmark edges with over 10 transitions are chosen for the evaluation. 2) Prediction on a road segment: We also test the performance of our method predicting traffic on road segments, using the Singapore traffic data.

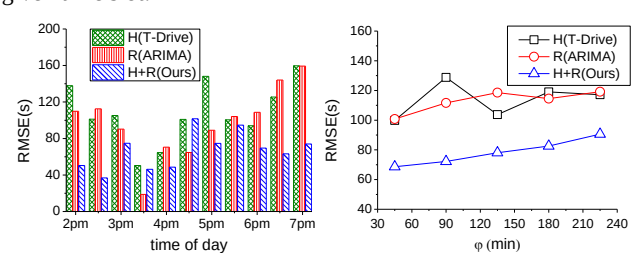
Baselines: We compare our $H + R$ approach with two baseline methods: 1) H method (T-Drive [26]). This method selects the travel time distribution from the historical traffic patterns according the time slot to be inferred, and then transfer the distribution into a travel time expectation. 2) R method (ARIMA [10], whose order is determined using AIC criterion). This is a well-known baseline method predicting the traffic conditions on a landmark edge (or a road segment) in terms of the samples (e.g., taxi trajectories) received a certain time (e. g, 1 or 2 hours) earlier than the time to be inferred.

Measurements: To quantify the accuracy of the traffic inference, we use the *root mean square error* (RMSE) defined as:

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_i (x_i - \hat{x}_i)^2} \tag{16}$$

where x_i is the real travel time, $\hat{x}_i$ is the predicted travel time and M is the number of predictions. Using this measurement, we study the performance of our approach changing over φ. If not specified, the default m is 2, i.e., second-order Markov Model.

Figure 11 a) shows the overall RMSE (the lower, the better) with φ =90min, τ=15min, between 2pm-7pm on weekdays. Clearly, $H + R$ outperforms both the H and R methods, especially in the rush hours (6pm-7pm), in which the traffic patterns (likely affected by multiple factors) change significantly and in complex ways; hence becomes difficult to predict for the baseline methods. Generally speaking, our method models a set of historical traffic patterns (for a landmark edge) conditioned by the recent traffic flows, $\mathrm{P}(H|R)$. Therefore, our method chooses different H patterns to predict future traffic in terms of the R. However, the stand alone H method only has one pattern corresponding to a given time slot.



a) w.r.t. time of day($\boldsymbol{\varphi}$=90min) b) w.r.t. $\boldsymbol{\varphi}$ ($\boldsymbol{\tau}$ =30min)

Figure 11: RMSE of different methods

Figure 11 b) plots the RMSE changing over φ with τ=30 minutes. As the delay φ increases, the performance of these approaches decreases while our approach has smaller RMSE (about a 30-second gap) than the competing methods. In short, given the same prediction error, our method is more capable of predicting traffic conditions at a farther time than H and R methods.

Figure 12 a) visualizes the distribution of residual error of the three methods, where $H + R$ has a clear advantage over H and R. Figure 13 investigates the performance varying in the order m, of our Markov model. Obviously, the 2nd-order model outperforms the 1st-order model because the traffic condition of a future time t depends on not only the state right before t but also a sequence of traffic states in the near past of t. However, the larger m we select, more data and heavier computation are needed. In practice, it is not necessary to choose a very large m given that the traffic condition does not rely on the distant past.

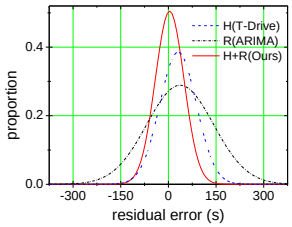


Figure 12: **Distribution of residual error**

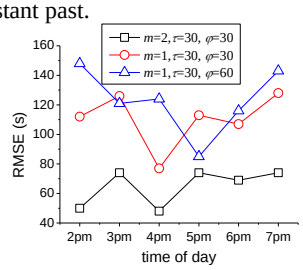


Figure 13. RMSE vs φ

Table 2 shows the RMSE of our method (with or without considering the weather) in predicting the future traffic conditions, using the time slot 6pm-7pm of the test days having a severe weather condition. Clearly, weather information brings significant benefit to our model. Given limited space we do not present more.

Table 2: RMSE considering weather information

φ (min)	with weather (s)	without weather (s)
30	**90.6**	106.6
60	**98.6**	107.1
90	**97.7**	140.4

Figure 14 shows the overall precision of the predictions on road segments using the Singapore traffic data. Here, the traveling speed of a road is discretized into four classes representing different volumes of traffic flows according to a pre-defined schema, e.g., green denotes >40km/h and yellow represents 20-40km/h. Though both ours and the H method outperform the R method, our approach did not show clear advantages over the H method according to the aggregated results (over 50 segments). But, our method does have a significantly better performance than the two baselines when predicting the traffic conditions on some road segments. So, we further explore these road segments, aiming to reveal the features (of roads) supporting our method.

Methods	Precision (over φ)	
	90min	120min
H(T-Drive)	0.686	0.689
R(ARIMA)	0.643	0.651
Ours	**0.683**	**0.688**

Figure 14: Overall precision of predictions on road segments

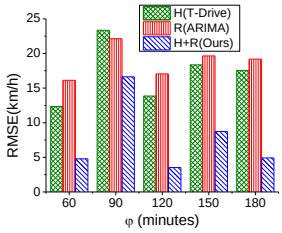
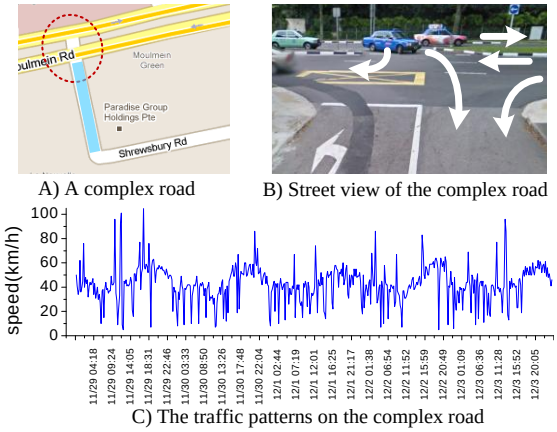


Figure 15: RMSE of the road

The Shrewsbury road, which is one of the good road segments (for ours), has a relatively complex linking structure in the road network, denoted as a blue segment in Figure 16 A). There are five kinds of directions and throughput that can happen at one of its terminal points (refer to the white arrows illustrated in Figure 16 B). Hence, the traffic pattern on this road becomes more complex and difficult to model. This claim is further justified by the traffic conditions plotted in Figure 16 C) where the travel speed is chaotic and disorderly over time of day. Neither H nor R can handle such a situation very well, and thus drops behind ours, as shown in Figure 15.

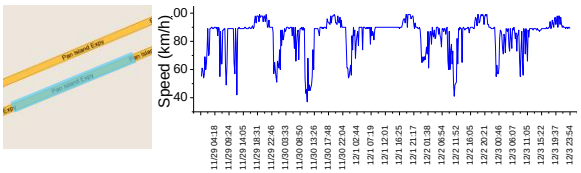


A) A complex road B) Street view of the complex road

C) The traffic patterns on the complex road

Figure 16: Traffic pattern study on Shrewsbury road

W.r.t. the road segments where our method is no better than the baselines, we found they have simple connecting structures in the road network. As demonstrated in Figure 17, Pan Island Expy is a straight road segment with only one link (to other road segments) at its terminal points (refer to Figure 17 A)). Thus, the traffic pattern on this road becomes similarly periodical and easy to predict (see Figure 17 B)) for the baselines. However, insufficient data (only 43 days) affects the precision of our model in inferring multiple $P(H|R)$.



A) A simple road B) The traffic patterns on the simple road

Figure 17: Traffic pattern study on Pan Island Expy road

As shown in Table 3, we further studied the average number of links connecting to a terminal point (node) in both the Beijing landmark graph and Singapore roads. Since a landmark (in Beijing data) usually has more links than a road segment (from Singapore), our method has a better overall performance on the Beijing dataset. Meanwhile, our method is more capable of predicting road segments with relatively more links to others. Given sufficient data, our method will show a higher performance.

Table 3: Average number of links at a terminal point

Datasets	On all segments	On good segments
Beijing landmark	3.1	8.7
Singapore	1.9	2.5

5.3 Evaluation on Routing

It is very difficult to *directly* evaluate whether a customized route (provided by our system) for a real user is the actually fastest one due to the following two reasons. First, a user can only drive on one route at a given time. You would never know if other routes are better (for the user) than the driven one. Requesting a different user to travel another route simultaneously would bring unexpected factors (caused by their different drive behaviors and

knowledge) to the evaluation. Second, it is not reasonable to request a single user to drive two different routes separately since the user can learn from their past driving experiences. So, the route driven later will benefit from the first test.

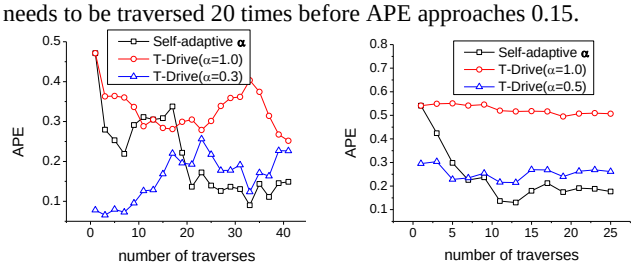
To address the above challenges, instead of directly finding the fastest driving route for a particular user, we first record the routes the user has driven with GPS logs and then estimate the travel time of these routes based on our method and baselines respectively, using the GPS logs as ground truths. More specifically, by mapping a route R to a landmark graph, we convert R into a sequence of landmark edges: $e_1, e_1, \ldots, e_n$. Then, we measure the accuracy of the estimation using the absolute percentage error (APE), defined as Equation (17).

$$\text{APE} = \frac{\sum_i^n |t(e_i) - \hat{t}(e_i)|}{\sum_i^n t(e_i)} \quad (17)$$

where $t(e_i)$ and $\hat{t}(e_i)$ are the real and predicted travel times of e_i.

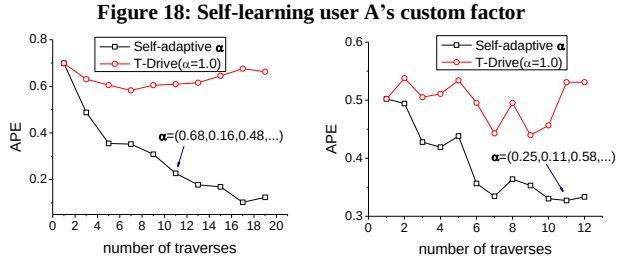
Here, we use two users' 1-year GPS logs (released in GeoLife dataset [28][29]) to determine the ground truth of the exact road segments a driver traversed and corresponding travel times. As proved in [26], T-Drive outperforms major routing services, such as speed-constraint-based and real-time-traffic-based methods, we only compare our approach with T-Drive here. Initially, we set α=1.0 on all the landmark edges for our method.

Figure 18 illustrates the self-adaptive process for learning user A's custom factor **α** on two different routes. First, the estimated travel time of our method gradually becomes accurate (measured by APE) and converges as the user A traverses these two routes more, showing advantage beyond that using a fixed **α**. For instance, when the user A traversed Route1 over 20 times, the APE of our method decreases to 0.15 while that of a fixed α = 1.0 is still 0.3. Choosing a small α = 0.3, T-Drive has a relatively minor APE in the first several days, however, it drops behind our method after user A has traveled the route several times. This is because a user's driving behavior changes over the times she has traveled a route. Second, the user A has different drive behaviors on these two routes. For example, our method can reach an APE of 0.15 after the user A traversed Route2 10 times while Route1 needs to be traversed 20 times before APE approaches 0.15.



(a) Route1 (b) Route2

Figure 18: Self-learning user A's custom factor



(a) User A (b) User B

Figure 19: Learning different two users' factors on the same route

Figure 19 plots the self-tuning processes of two users traversing the same route, demonstrating the fact that different users have different custom factors tuned in different ways. For instance, we see the clear difference between these two users' **α** after they traversed the route 11 times. Note that **α** is a vector rather than a single value and a route could include several landmark edges. According to these results, we should neither use the same custom factor for different users nor set a consistent factor for a particular user on different routes. Additionally, the custom factor of a user is dynamic and changes over the user's driving experiences and skills on a road. So, our self-adaptive routing out-performs T-Drive which is better than other major services.

Efficiency: Besides being effective, our system is also efficient due to the following reasons. First, a landmark graph is only a subset of the original road network (8% in node size, 16% in edge size). So, the rough routing on the landmark graph is very fast. Also, a rough route indicating the key directions reduces the search area on a road network and enables parallel computing when performing the detailed routing [26]. Second, the high dimensional embedding approach speeds up the traffic prediction tremendously, as depicted in Figure 20. After calculating the 1-step transition matrix which has the same computation with the statistic-based approach, our method computes the h-step ($h \geq 2$) very efficiently. The time cost shown in Figure 21 is an average time on calculating six transitions, h =1-6. Third, we only include the items (about 0.1% of |**α**| according to a study) with significant changes, sending a query to the *Cloud*. To reveal the performance of our method (regardless of system design), we test our system on a single server with 2.67GHz CPU and 16GB RAM (using a single thread without optimization) in the *Cloud*, as shown in Figure 21. The mobile client is running on a Windows smartphone with 1GHz CPU and GPRS connection. Roughly, we can answer 1,000 queries per second using 30 (24-core) servers in a Cloud.

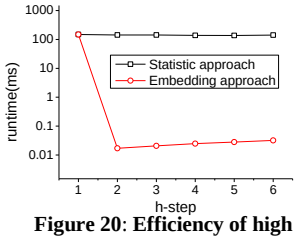


Figure 20: Efficiency of high dimensional embedding

Mobile	T	Cloud	T
Query send	144 ms	Predict traffic	25 ms
Results down	287 ms	Route compute	326 ms
Learning new **α**	20 us	Build landmark	6h

Figure 21: Time cost study on operations of our system

6. RELATED WORK

6.1 Traffic Estimation

There are a few projects [1][2][9][11][12][25] aiming to learn historical traffic patterns, estimate real-time traffic flows and forecast future traffic conditions on some road segments in terms of floating car data [23], such as GPS trajectories as well as Wi-Fi and GSM signals. However, these methods are road-segment-level inferences, which predict the traffic conditions on individual road segments with enough samples. Although our prediction model can also be used on a road segment, our work differs from the above methods in the following aspects. First, by using the landmark graphs, our work well models the city-wide traffic conditions from low sampling-rate trajectories (e.g., 3-5minutes per sample), and enables a real routing service. Second, the routing service considers the driver behavior both of an end user (for whom the route is being computed) and taxi drivers. Third, according to the experimental results our prediction model outperforms competing methods, such as ARIMA [10].

6.2 Smart Routing

To optimize taxi drivers' income, literatures [7][19] has proposed route recommendation services for a taxi driver by analyzing fleet trajectories. Here, they focus on taxi drivers' pick-up behavior in creating higher profit (e.g., how to easily find passengers). So, a

normal end user cannot benefit from the recommended routes. Meanwhile, the traffic conditions are not involved in their systems.

Papers like [17][18][21] present work that aims to provide personalized routes according to a user's driving preferences in choosing a road, using user-computer interaction or implicit modeling. The recommended routes from these works are not optimized by travel time. Different from these works, the route we recommend to a driver is the practically fastest one customized for a particular driver, considering both traffic conditions of a future time (when the computed route will be actually traversed) and the behavior of the driver. Other factors, like day of the week, and weather conditions, are also considered in our routing model.

Existing work [6][8][26] also aims to use user-generated GPS trajectories to improve routing services. The work presented in this paper significantly differs from those examples, especially our previous publication T-Drive [26] in the following five aspects. 1) T-Drive does not consider the drive behavior of end users for whom a route is computed. Also, a user's mobile phone does not provide any knowledge of the user. But, our method self-learns different users' drive behaviors (varying in different roads and the times they traversed a road) according to the recent GPS logs, automatically and gradually, in their own mobile phones. The interaction between *mobile* and *Cloud* enables us to find the *practically* fastest driving route *customized* for a user. 2) T-Drive only employs the historical traffic patterns in the routing process. However, we infer the traffic conditions at a future time (when a road is actually driven) based on the historical patterns and real-time traffic flow. The future traffic conditions are involved in the routing. 3) We incorporate other resources from the Web, like weather condition records, into the routing model. 4) We differentiate different taxi drivers' knowledge in different regions. This helps us better model the traffic patterns and taxi drivers' knowledge in choosing a route. 5) Given the above differences, we estimate the travel time of a route more precisely than T-Drive according to the extensive evaluations; our method hence can find better driving routes for a particular user.

7. CONCLUSION

This paper describes a system for computing shortest-time driving routes using traffic information and driver behavior. Specifically, the system mines historical traffic patterns (from GPS trajectories generated by taxicabs) and incorporates recent real-time traffic information (from the same fleet or road sensors) to predict future traffic conditions at the time when the computed route is actually driven. The system incorporates day of the week, time of day, weather conditions, and individual driving strategies (both of the fleet drivers and of the end user for whom the route is being computed). We build our system with a real-world dataset generated by over 33,000 taxis in Beijing, and evaluate our services with extensive experiments and in-the-field studies. The prediction model is also tested using a Singapore traffic dataset. The results show that: 1) our prediction method considering both historical patterns and real-time traffic, $H + R$, outperforms the approaches separately using H and R in predicting the future traffic conditions, especially, in handling road segments with relatively more links (to other segments). The proposed high dimensional embedding method speeds up the 2nd-order Markov model and enables the online traffic prediction. 2) Our system accurately estimates the travel time of a driving route for a particular user by self-tuning the custom factors (on different roads) for the user in terms of the user's historical GPS logs. So, we can find *practically* fast routes *customized* for a particular user.

In the future, we plan to learn a user's driver behavior in a mobile phone, with a more efficient, accurate and advanced method.

8. REFERENCE

[1] Bejan, A., I., Gibbens, R., J., Evans D., Beresford, A., R., Bacon, J., Friday A. Statistical Modelling and Analysis of Sparse Bus Probe Data in Urban Areas, In Proc. ITS, 2010.

[2] Castro-Neto, M., Jeong, Y. S., Jeong, M., K., Han, L., D. Online-SVR for short-term traffic prediction under typical and atypical traffic conditions. Expert systems with applications. 36, 2009

[3] Chhikara, R., S., inverse L. F. L. The inverse Gaussian distribution: theory, methodology, and applications, 1989.

[4] Coppersmith, D., Winograd S. Matrix multiplication via arithmetic progressions. Journal of symbolic computation; 1990:251--280.

[5] Ding, B., Yu, J. X., Qin, L. Finding time-dependent shortest paths over large graphs. In Proc. EDBT; 2008:205-216.

[6] Fawcett, J. and P. Robinson, Adaptive Routing for Road Traffic. IEEE Computer Graphics and Applications, 2000. 20(3): p. 46-53.

[7] Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser M., Pazzani M. J. An Energy-Efficient Mobile Recommender System. In Proc. KDD 2010.

[8] Gonzalez H., Han J., Li X., Myslinska M., Sondag J. P. Adaptive fastest path computation on a road network: A traffic mining approach. In Proc. VLDB, 2007.

[9] Guehnemann A., Schaefer R. P., Thiessenhusen K. U., Wagner P. Monitoring traffic and emissions by floating car data. Institute of transport studies Australia, 2004.

[10] Hamilton J. Time series analysis: Princeton University Press, 1994.

[11] Herrera, J. C., Work, D. Ban, X., Herring, R. Jacobson, Q. and Bayen, A. Evaluation of traffic data obtained via GPS-enabled mobile phones: the *Mobile Century* field experiment. *Transportation Research C*, 18, pp. 568–583, 2010.

[12] Herring R., Hofleitner A., Abbeel P., Bayen A. Estimating arterial traffic conditions using sparse probe data. In Proc. ITS, 2010.

[13] Hunter T., Herring R., Abbeel P., Bayen A. Path and travel time inference from GPS probe vehicle data. In Proc. Neural Information Processing Systems foundation (NIPS), 2009.

[14] Kanoulas E., Du Y., Xia T., D. Z. Finding fastest paths on a road network with speed patterns. In Proc. ICDE, 2006.

[15] Karlin S., Taylor H. M. A first course in stochastic processes, 1975.

[16] Leibowitz, N., Baum, B., Enden, G., Karniel, A. The exponential learning equation as a function of successful trials results in sigmoid performance, Journal of Mathematical Psychology, 54(3):338-340, 2010.

[17] Letchner J., Krumm J., and Horvitz E., Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning, In Proc. IAAI, 2006.

[18] Liu B. Route Finding by using knowledge about the road network. IEEE Trans. on systems, man and cybernetics. 27 (4), 1997.

[19] Liu, L., Andris, C., Biderman, A., Ratti, C. Uncovering cabdrivers' behavior patterns from their digital traces. Computers, Environment and Urban Systems, 2010.

[20] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Huang, Y. Map-matching for low-sampling-rate GPS trajectories. In Proc. ACM SIGSPATIAL GIS, 2009.

[21] McGinty, L. and Smyth, B. Turas: A Personalized Route Planning System. In Proc. Sixth Pacific Rim International Conference on AI, PRICAI, 2000.

[22] Orda, A., Rom R. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. JACM; 1990:625.

[23] Pfoser D. Floating Car Data. Encyclopedia of GIS 2008

[24] Raftery A. E. A model for high-order Markov chains. Journal of the Royal Statistical Society. Series B (Methodological); 1985:528--39.

[25] Thiagarajan A., et al. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems, 2009.

[26] Yuan J., Zheng Y., Zhang C. Y., Xie W., Xie, X., Sun, G., Huang, Y. T-Drive: Driving Directions Based on Taxi Trajectories. In Proc. ACM SIGSPATIAL GIS, 2010.

[27] Yuan J., Zheng Y., Zhang C. Y., Xie X. An Interactive-Voting based Map Matching Algorithm. In Proc. MDM, 2010.

[28] Zheng, Y., Chen, Y., Xie, X., Ma, W., Y. GeoLife2.0: A Location-Based Social Networking Service. In Proc. MDM 2009.

[29] Zheng, Y., Xie, X., Ma, W., Y. GeoLife: A Collaborative Social Network-ing Service among user, location and trajectory. IEEE Data Engineering Bulletin, 2010. 33(2), 2010, pp. 32-40