T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence

Jing Yuan, *Student Member*, *IEEE*, Yu Zheng, *Senior Member*, *IEEE*, Xing Xie, *Senior Member*, *IEEE*, and Guangzhong Sun, *Member*, *IEEE*

Abstract—This paper presents a smart driving direction system leveraging the intelligence of experienced drivers. In this system, GPS-equipped taxis are employed as mobile sensors probing the traffic rhythm of a city and taxi drivers' intelligence in choosing driving directions in the physical world. We propose a time-dependent landmark graph to model the dynamic traffic pattern as well as the intelligence of experienced drivers so as to provide a user with the practically fastest route to a given destination at a given departure time. Then, a Variance-Entropy-Based Clustering approach is devised to estimate the distribution of travel time between two landmarks in different time slots. Based on this graph, we design a two-stage routing algorithm to compute the practically fastest and customized route for end users. We build our system based on a real-world trajectory data set generated by over 33,000 taxis in a period of three months, and evaluate the system by conducting both synthetic experiments and in-the-field evaluations. As a result, 60-70 percent of the routes suggested by our method are faster than the competing methods, and 20 percent of the routes share the same results. On average, 50 percent of our routes are at least 20 percent faster than the competing approaches.

Index Terms—Spatial databases and GIS, data mining, GPS trajectory, driving directions, driving behavior

1 Introduction

Finding efficient driving directions has become a daily activity and been implemented as a key feature in many map services like Google and Bing Maps. A fast driving route saves not only the time of a driver but also energy consumption (as most gas is wasted in traffic jams). Therefore, this service is important for both end users and governments aiming to ease traffic problems and protect environment.

Essentially, the time that a driver traverses a route depends on the following three aspects: 1) the physical feature of a route, such as distance, capacity (lanes), and the number of traffic lights as well as direction turns; 2) the time-dependent traffic flow on the route; and 3) a user's driving behavior. Given the same route, cautious drivers will likely drive relatively slower than those preferring driving very fast and aggressively. Also, users' driving behaviors usually vary in their progressing driving experiences. For example, 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/fastest path computing.

Usually, big cities have a large number of taxicabs traversing in urban areas. For efficient taxi dispatching and monitoring, taxis are usually equipped with a GPS sensor, which enables them to report their locations to a server at regular intervals, e.g., 2-3 minutes. That is, a lot of GPS-equipped taxis already exist in major cities, generating a huge number of GPS trajectories every day [2]. Intuitively, taxi drivers are experienced drivers who can usually find out the fastest route to send passengers to a destination based on their knowledge (we believe most taxi drivers are honest although a few of them might give passengers a roundabout trip). When selecting driving directions, besides the distance of a route, they also consider other factors, such as the time-variant traffic flows on road surfaces, traffic signals and direction changes contained in a route. These factors can be learned by experienced drivers but are too subtle and difficult to incorporate into existing routing engines. Therefore, these historical taxi trajectories, which imply the intelligence of experienced drivers, provide us with a valuable resource to learn practically fast driving directions.

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. As shown in Fig. 1, 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 Internet, like web maps and 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. Finally, 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

- *J. Yuan and G. Sun are with the University of Science and Technology of China, No. 96 Jinzhai road, Hefei, Anhui 230026, China. E-mail: yuanjing@mail.ustc.edu.cn, gzsun@ustc.edu.cn.*
- *Y. Zheng and X. Xie are with Microsoft Research Asia, No. 5 Danling Street, Haidian district, Beijing 100080, China. E-mail: {yuzheng, xing.xie}@microsoft.com.*

Manuscript received 8 Mar. 2011; revised 17 July 2011; accepted 19 Aug. 2011; published online 19 Sept. 2011.
Recommended for acceptance by X. Zhou.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-03-0111.
Digital Object Identifier no. 10.1109/TKDE.2011.200.

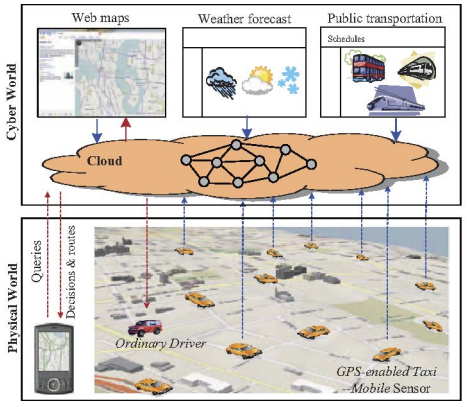


Fig. 1. A cloud-based driving directions service.

driving routes (recorded in GPS logs), and supports the cloud to customize a practically fastest route for the user.

However, we need to face the following three challenges: 1) *Intelligence modeling.* As a user can select any place as a source or destination, there would be no taxi trajectory exactly passing the query points. That is, we cannot answer user queries by directly mining trajectory patterns from the data. Therefore, how to model taxi drivers' intelligence that can answer a variety of queries is a challenge; 2) *Data sparseness and coverage.* We cannot guarantee there are sufficient taxis traversing on each road segment even if we have a large number of taxis. That is, we cannot accurately estimate the speed pattern of each road segment; and 3) *Low-sampling-rate problem.* 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 [3]. As shown in Fig. 2, there could exist four possible routes (R_1-R_4) traversing the sampling points a and b.

Since this paper is an extension of our previous publication [1], we summarize the contributions (including that of the previous paper) of our work as follows:

1. In the previous paper [1], we propose the notion of a time-dependent landmark graph, which well models the intelligence of taxi drivers based on the taxi trajectories. We devise a Variance-Entropy-Based Clustering (VE-Clustering for short) method to learn the time-variant distributions of the travel times between any two landmarks.
2. In this extension work:
   a. We further improve our routing service by self-adaptively learning the driving behaviors of both the taxi drivers and the end users so as to provide personalized routes to the users.
   b. We present smoothing algorithms for removing the roundabout part of the original rough routes.
   c. We build the improved system by using a real-world trajectory data set generated by 33,000+ taxis in a period of three months, and evaluate the system by conducting both synthetic experiments and in-the-field evaluations (performed by real drivers). The results show that proposed method can effectively and efficiently find out practically better routes than the competing methods.

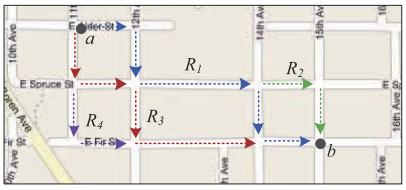


Fig. 2. Low-sampling-rate problem.

2 Preliminary

In this section, we first introduce some terms used in this paper, then define our problem.

Definition 2.1 (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 2.2 (Road network). *A* road network G_r *is a directed graph, $G_r = (V_r, E_r)$, where V_r is a set of nodes representing the terminal points of road segments, and E_r is a set of edges denoting road segments. The time needed for traversing an edge is dynamic during time of day.*

Definition 2.3 (Route). *A* route R *is a set of connected road segments, i.e., $R : r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$, where $r_{k+1}.s = r_k.e$, $(1 \leq k < n)$. The start point and end point of a route can be represented as $R.s = r_1.s$ and $R.e = r_n.e$.*

Definition 2.4 (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 time stamp $p.t$, i.e., $Tr : p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, where $0 < p_{i+1}.t - p_i.t < \triangle T$ $(1 \leq i < n)$. $\triangle T$ defines the maximum sampling interval between two consecutive GPS points.*

3 Time-Dependent Landmark Graph

This section first describes the construction of the time-dependent landmark graph, and then details the travel time estimation of landmark edges.

3.1 Building the Landmark Graph

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 [3], [4]. Meanwhile, we cannot guarantee 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. There is a tag associated with a taxi's reporting when the taximeter is turn on or off, i.e., a passenger get on or off the taxi. Then, we employ our IVMM algorithm [4], which has better performance than existing map-matching algorithms when dealing with the low-sampling rate trajectories. This algorithm

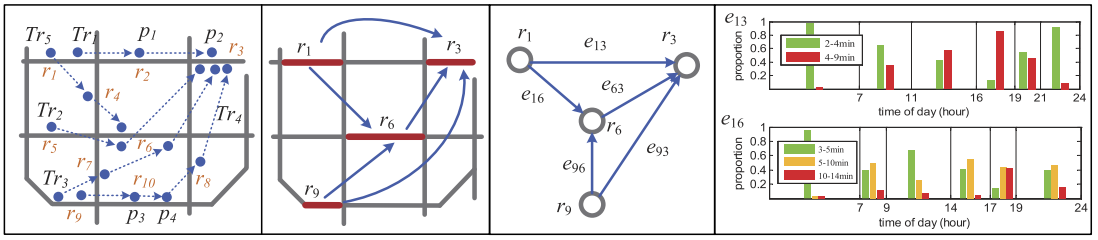


Fig. 3. Landmark graph construction.

utilizes the spatial-temporal restrictions to obtain candidate road segments, then considers the mutual influences of the GPS points in a trajectory to calculate static/dynamic score matrix for a trajectory and performs a voting-based approach among all the candidates. As a result, each taxi trajectory is converted to a sequence of road segments. We formally define the *landmark* as follows:

Definition 3.1 (Landmark). *A* landmark *is one of the top-k road segments that are frequently traversed by taxi drivers according to the trajectory archive.*

Based on the preprocessed taxi trajectories, we detect the top-k frequently traversed road segments, which are termed as landmarks. The reason why we use "landmark" to model the taxi drivers' intelligence is that: 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. Later, we connect two landmarks according to Definitions 3.2, 3.3, and 3.4.

Definition 3.2 (Transition). *Given a trajectory archive A, 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:*

1. *There exists a trajectory $T_r = (p_1, p_2, \ldots, p_n) \in A$, after map matching, T_r 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. $u = r_i, v = r_j$.*
2. *$r_{i+1}, r_{i+2}, \ldots, r_{j-1}$ are not landmarks.*
3. *$t_a = p_i.t, t_l = p_j.t$ and the* travel time *of this transition is $t_l - t_a \leq t_{max}$.*

Definition 3.3 (Candidate edge and frequency). *Given two landmarks u, v and the trajectory archive A, 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 support *of e, denoted as $e.supp$, is the number of transitions connecting (u, v), i.e., $|S_{uv}|$. The* frequency *of e is $e.supp/\tau$, denoted as $e.freq$, where τ represents the total duration of trajectories in archive A.*

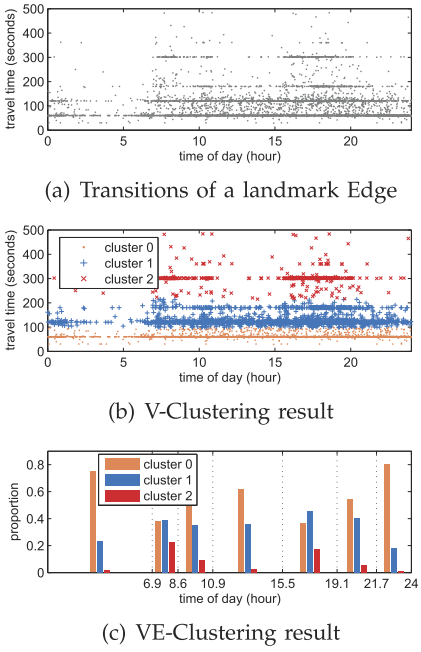
Definition 3.4 (Landmark Edge). *Given a candidate edge e and a minimum frequency threshold δ, we say e is a landmark edge if $e.freq \geq \delta$.*

Definition 3.5 (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 conditioned by δ and t_{max}.*

The threshold δ is used to eliminate the edges seldom traversed by taxis, as the fewer taxis that pass two landmarks, the lower accuracy of the estimated travel time (between the two landmarks) could be. Additionally, we set the t_{max} value to remove the landmark edges having a very long travel time. Due to the low-sampling rate problem, sometimes, a taxi may consecutively traverse three landmarks while no point is recorded when passing the middle (second) one. This will result in that the travel time between the first and third landmark is very long. Such kinds of edges would not only increase the space complexity of a landmark graph but also bring inaccuracy to the travel time estimation (as a farther distance between landmarks leads to a higher uncertainty of the traversed routes). We use the frequency instead of the support of a landmark edge (to guarantee efficient transitions) because we want to eliminate the effect induced by the scale of the trajectory archive.

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 patterns. Therefore, we build two different landmark graphs for weekdays and weekends, respectively. That is, we project all the weekday trajectories (from different weeks and months) into one weekday landmark graph, and put all the weekend trajectories into the weekend landmark graph. 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, $2 \times 2 = 4$ landmark graphs are built. The weather condition records are crawled from the weather forecast website.

Figs. 3A, 3B, and 3C illustrate an example of building the 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 (T_{r4}) can only be counted once



(a) Transitions of a landmark Edge

(b) V-Clustering result

(c) VE-Clustering result

Fig. 4. An example of VE-Clustering Algorithm.

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 (although the taxi driver only traversed the segment once), as shown in Fig. 3C. After the detection of landmarks, we convert each taxi trajectory from a sequence of road segments to a landmark sequence, and then connect two landmarks with an edge if the transitions between these two landmarks conform to Definition 3.4 (supposing $\delta = 1$ in this example).

3.2 Travel Time Estimation

In this step, we aim to automatically partition time of a day into several slots (for different landmark edges) (see Fig. 4c) according to the traffic conditions reflected by the raw samples (as shown in Fig. 4a) pertaining to a landmark edge. Then, we estimate the travel time distribution of each time slot for each landmark edge.

3.2.1 VE-Clustering

Since the road network is dynamic (refer to Definition 2.2), we can use neither the same nor a predefined time partition method for all the landmark edges. Meanwhile, as shown in Fig. 4a, the travel times of transitions pertaining to a landmark edge clearly gather around some values (like a set of clusters) rather than a single value or a typical Gaussian distribution, as many people expected. 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, skill and preferences. Therefore, different from existing methods [5], [6] regarding the travel time of an edge as a single-valued function based on time of day, we consider a landmark edge's travel time as a set of distributions corresponding to different time slots. Additionally, the distributions of different edges, such as e_{13} and e_{16}, change differently over time.

To address this issue, we develop the VE-Clustering algorithm (refer to [1] for the pseudocode), which is a two-phase clustering method, to learn different time partitions for different landmark edges based on the taxi trajectories. In the first phase, called V-clustering, we cluster the travel times of transitions pertaining to a landmark edge into several categories based on the variance of these transitions' travel times. In the second phase, termed E-clustering, we employ the information gain to automatically learn a proper time partition for each landmark edge. Later, we can estimate the distributions of travel times in different time slots of each landmark edge.

The reason why we conduct the following V-Clustering instead of using some k-means-like algorithm or a predefined partition is that the number of clusters and the boundaries of these clusters vary in different landmark edges.

V-Clustering. We first sort $\mathcal{T}_{uv}$ according to the values of travel time $(t_l - t_a)$, and then partition the sorted list L into several sublists in a binary-recursive way. In each iteration, we first compute the variance of all the travel times in L. Later, we find the "best" split point having the minimal weighted average variance (WAV) defined as

$$\mathrm{WAV}(i; L) = \frac{|L_A^{(i)}|}{|L|}\mathbb{W}(L_A^{(i)}) + \frac{|L_B^{(i)}|}{|L|}\mathbb{W}(L_B^{(i)}), \tag{1}$$

where $L_A^{(i)}$ and $L_B^{(i)}$ are two sublists of L split at the ith element and $\mathbb{W}$ represents the variance. This best split point leads to a maximum decrease of

$$\triangle\mathbb{W}^{(i)}(L) = \mathbb{W}(L) - \mathrm{WAV}(i; L). \tag{2}$$

The algorithm terminates when $\max_i\{\triangle\mathbb{W}(i)\}$ is less than a threshold (this will definitely happen due to Theorem 3.6, refer to the appendix part, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.200, for the strict proof). As a result, we can find out a set of split points dividing the whole list L into several clusters $C = \{c_1, c_2, \ldots, c_m\}$, each of which represents a category of travel times. As shown in Fig. 4b, the travel times of the landmark edges have been clustered into three categories plotted in different colors and symbols.

Theorem 3.6. *$L = \{x_i\}_{i=1}^N$ is a sorted list, denote $L_A^{(i)} = \{x_j\}_{j=1}^i$ and $L_B^{(i)} = \{x_j\}_{j=i+1}^N$, let*

$$\triangle\mathbb{W}^{(i)}(L) = \mathbb{W}(L) - \frac{|L_B^{(i)}|\mathbb{W}(L_B^{(i)}) + |L_B^{(i)}|\mathbb{W}(L_B^{(i)})}{|L|}.$$

If $\triangle\mathbb{W}(L) = \max_i\{\triangle\mathbb{W}^{(i)}(L)\}$, then $\triangle\mathbb{W}(L)|L| \geq \triangle\mathbb{W}(L_A^{(i)})|L_A^{(i)}|$ and $\triangle\mathbb{W}(L)|L| \geq \triangle\mathbb{W}(L_B^{(i)})|L_B^{(i)}|$ for $\forall i = 1, 2, \ldots, N$, the equality holds only if $\triangle\mathbb{W}(L_A^{(i)}) = 0$ and $\triangle\mathbb{W}(L_B^{(i)}) = 0$, respectively.

E-Clustering. This step aims to split the x-axis into several time slots such that the travel times have a relatively stable distribution in each slot. After V-Clustering, we can represent each travel time y_i with the category it pertains to $(c(y_i))$, and then sort the pair collection $S^{xc} = \{(x_i, c(y_i))\}_{i=1}^n$ according to x_i (arriving time). The information entropy of the collection S^{xc} is given by

$$\mathrm{Ent}(S^{xc}) = -\sum_{i=1}^{m} p_i \log(p_i), \quad (3)$$

where p_i is the proportion of a category c_i in the collection. The E-Clustering algorithm runs in a similar way to the V-Clustering to iteratively find out a set of split points. The only difference between them is that, instead of the WAV, we use the weighted average entropy of S^{xc} defined as

$$\mathrm{WAE}(i; S^{xc}) = \frac{|S_1^{xc}(i)|}{|S^{xc}|}\mathrm{Ent}(S_1^{xc}(i)) + \frac{|S_2^{xc}(i)|}{|S^{xc}|}\mathrm{Ent}\big(S_2^{xc}(i)\big)$$

in the E-Clustering, where S_1^{xc} and S_2^{xc} are two subsets of S^{xc} when split at the ith pair. The best split point induces a maximum information gain which is given by

$$\triangle E(i) = \mathrm{Ent}(S^{xc}) - \mathrm{WAE}\big(i; S^{xc}\big).$$

As demonstrated in Fig. 4c, we can compute the distribution of the travel times in each time slot after the E-Clustering process.

3.2.2 Differentiate Taxi Drivers' Experiences

For a big city like New York and Beijing, not all the taxi drivers are familiar with the traffic flows of the whole city. According to the learning theory, typically, the taxi driver's knowledge of the traffic flow in a ceratin area of a city will grow with the cumulative number that he traveled to that area. Suppose a landmark edge e_{uv} was traversed by n different taxi drivers. The transition set S_{uv} can be accordingly categories into n sample spaces. After VE-Clustering, the time during a day is partitioned into several time slots. Let D_i be the travel time distribution during a certain time slot using only the sample from taxi driver i, denoted as

$$\begin{pmatrix} 1 & 2 & \dots & k \\ p_1^i & p_2^i & \dots & p_k^i \end{pmatrix}, \quad (4)$$

where $1, 2, \dots, k$ stand for k different travel time clusters of this landmark edge and $p_1^i, p_2^i \dots, p_k^i$ represent the proportion based on the sample space of taxi driver i. The growing of the familiarity is modeled using a Sigmoid learning curve [7], defined as

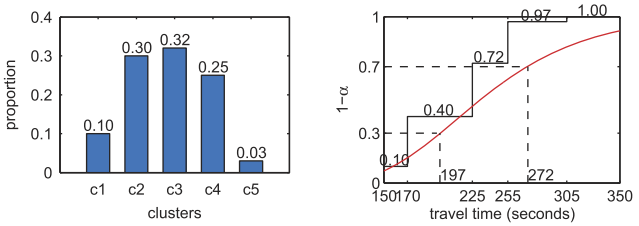
$$f(n_i) = \frac{1}{1 + e^{-(an_i + b)}}, \quad (5)$$

where $f(n_i)$ is the familiarity, a, b are the coefficients, and n_i is the number of times traversed by taxi driver i. $an_i + b$ is the linear transformation which maps n_i from $[\min, \max]$ to $[-6, 6]$, where min and max are the minimum number of transitions and maximum number of transition on this landmark edge, respectively. Then, the refined distribution of this time slot, denoted by D, is computed by the weighted average

$$\begin{pmatrix} 1 & 2 & \dots & k \\ \sum w_i p_1^i & \sum w_i p_2^i & \dots & \sum w_i p_k^i \end{pmatrix}, \quad (6)$$

where the weight w_i is the normalized familiarity of taxi driver i

$$w_i = \frac{f(n_i)}{\sum_{i=1}^{n} f(n_i)}. \quad (7)$$



(a) Travel time distribution (b) Cumulative frequency

Fig. 5. Travel time w.r.t. custom factor.

4 Route Computing

This section introduces the routing algorithm, which consists of two stages: rough routing in the landmark graph and refined routing in the real road network.

4.1 Rough Routing

4.1.1 Rough Route Generation

Besides the traffic condition of a road, the travel time of a route also depends on drivers. Sometimes, different drivers take different amounts of time to traverse the same route at the same time slot. The reasons lie in a driver's driving habit, skills and familiarity of routes. For example, people familiar with a route can usually pass the route faster than a newcomer. Also, even on the same path, cautious people will likely drive relatively slower than those preferring to drive very fast and aggressively. To catch the above factor caused by individual drivers, we define the *custom factor* as follows:

Definition 4.1 (Custom Factor). *The custom factor α indicates how fast a person would like to drive as compared to taxi drivers. The higher rank (position in taxi drivers), the faster the person would like to drive.*

For example, $\alpha = 0.7$ means that you can outperform 70 percent taxi drivers in terms of travel time under the same external conditions (traffic flow, signal, weather, etc.). Initially, we set a default value for different users. Later, in Section 4.3, we will detail our approach for learning the custom factor for each user in a self-adaptive way with the continuous use of our service and providing a personalized route for different users.

Given a user's custom factor α, we can determine his/her time cost for traversing a landmark edge e in each time slot based on the learned travel time distribution. For example, Fig. 5a depicts the travel time distribution of a landmark edge in a given time slot (c_1-c_5 denotes five categories of travel times). Then, we convert this distribution into a cumulative frequency distribution function and fit a continuous cumulative frequency curve shown in Fig. 5b. Note this curve represents the distribution of travel time in a given time slot. That is, the travel times of different drivers in the same time slot are different. So, we cannot use a single-valued function. For example, given $\alpha = 0.7$, we can find out the corresponding travel time is 272 seconds, while if we set $\alpha = 0.3$ the travel time becomes 197 seconds.

Now, the rough routing problem becomes the typical *time-dependent fastest path* (TDFF) problem. The complexity of solving this problem depends on whether the network satisfies the "FIFO" (first in, first out) property "In a network

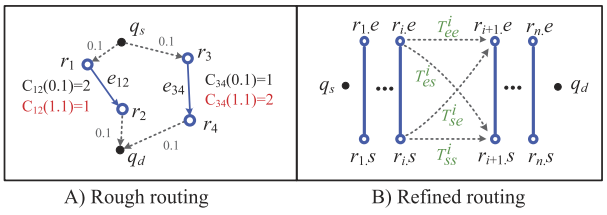


Fig. 6. Rough routing and refined routing.

$G = (V, E)$, if A leaves node u starting at time t_1 and B leaves node u at time $t_2 \geq t_1$, then B cannot arrive at v before A for any arc (u,v) in E." In practise, many networks, particularly transportation networks, exhibit this behavior [8]. If a driver's route spans more than one time slot, we use can refine the travel time cost to be FIFO (refer to Appendix, available in the online supplemental material).

In the rough routing, we first search m (in our system, we set $m = 3$) nearest landmarks for q_s and q_d, respectively (a spatial index is used), and formulate $m \times m$ pair of landmarks. For each pair of landmarks, we find the time-dependent fastest route on the landmark graph by using the Label-Setting algorithm [8], which is a generalization of Dijkstra algorithm. For any visited landmark edge, we use the custom factor to determine the travel time. The time costs for traveling from q_s and q_e to their nearest landmarks are estimated in terms of speed constraint.

For example, in Fig. 6A, if we start at time $t_d = 0$, the fastest route from q_s to q_d is $q_s \rightarrow r_3 \rightarrow r_4 \rightarrow q_d$. When we arrive at r_3, the time stamp is 0.1, the travel time of e_{34} is 1, then the total time of this route is $0.1 + 1 + 0.1 = 1.2$. However, if we start at $t_d = 1$, the route $q_s \rightarrow r_1 \rightarrow r_2 \rightarrow q_d$ now becomes the fastest rough route since when we arrive at r_3, the travel time of the e_{34} becomes 2 and the total time of the previous route is now 2.2.

4.1.2 Rough Route Smoothing

Even using the state-of-the-art map-matching algorithm, the accuracy is less than 70 percent [4] for the low-sampling rate trajectories. For example, as shown in Fig. 7, r_2 and r_4 are wrongly mapped road segments, the actual route is along the horizontal road from q_s to q_d. The map-matching error results in that r_2 and r_4 are recognized as landmarks and brings noise when estimating the travel time, e.g., the real travel time for $r_2 \rightarrow r_3$ is very likely to be much longer than the estimated time due to the map-matching error, which leads to $r_2 \rightarrow r_3$ becomes a part of this rough route.

Let $q_s \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow \cdots \rightarrow l_{n-1} \rightarrow l_n \rightarrow q_d$ be the rough route computed based on Section 4.1, where each l_i is a landmark ($i = 1, 2, \ldots, n$). We present a postprocessing to smooth the roundabout rough route. We summarize three key characteristics of a nonroundabout route, termed as *Nonroundabout Principles*. Given a rough route $R_{rough} : q_s \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow \cdots \rightarrow l_{n-1} \rightarrow l_n \rightarrow q_d$, we say R_{rough} satisfies

1. *Source-Farther Principle* if $\forall i = 1, 2, \ldots n - 1$, $\text{dist}(l_{i+1,q_s}) > \text{dist}(l_i, q_s)$,
2. *Destination-Closer Principle* if $\forall i = 1, 2, \ldots n - 1$, $\text{dist}(l_{i+1}, q_d) < \text{dist}(l_i, q_d)$, and
3. *Next-Nearest Principle* if $\forall i = 1, 2, \ldots n - 1, \text{dist}(l_i, l_{i+1}) = \min_{j>i}\{\text{dist}(l_i, l_j)\}$,

where $\text{dist}(l_i, l_j)$ is the road network distance from l_i to l_j.

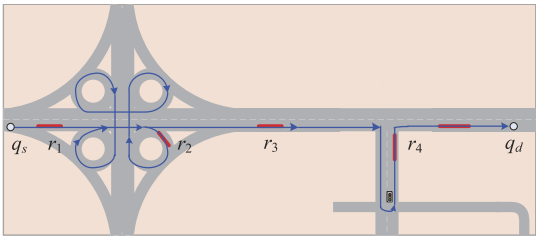


Fig. 7. An example of a roundabout rough route.

The source-farther principles states that the distance from each landmark to the source should be farther than its previous one. The destination-closer principles states that each landmark should be closer than its previous landmark to the destination. As shown in Fig. 7, r_2 violates the Principle 2. Principles 1 and 2 mean that each step ahead should have contribution to this trip. The next-nearest principle (also termed as "nonturn-back" principle) states that the next landmark l_{i+1} should be the nearest landmark of l_i among all the landmarks after l_i. Otherwise, we can go directly from l_i to its next nearest landmark to avoid the "U-turn" (refer to r_4 in Fig. 7 as an example). Based on the nonroundabout principles, we define the *roundabout rough route* as follows:

Definition 4.2 (Roundabout Rough Route). *A rough route R_{rough} is called a* roundabout rough route *if it violates any of the three nonroundabout principles.*

Definition 4.3 (Smoothing Problem). *Given a roundabout rough route R_{rough}, we aim to extract a nonroundabout rough route with the minimum loss of information (which could be obtained from the original landmarks).*

We first pick up the longest landmark subsequence based on R_{rough} that satisfies Principles 1 and 2 (termed as *global smoothing*), then rebuild a rough route according to Principle 3 (termed as *local smoothing*).

Global Smoothing. Suppose $R_{rough} : q_s \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow \cdots \rightarrow l_{n-1} \rightarrow l_n \rightarrow q_d$ is a roundabout rough route. The problem of finding a longest landmark sequence obeying Principle 2 is equivalent to finding a *longest increasing subsequence* from the sequence $(\text{dist}(l_1, q_s), \text{dist}(l_2, q_s), \ldots, \text{dist}(l_n, q_s))$. The longest increasing subsequence problem can be solved in time $O(n \log n)$ using the

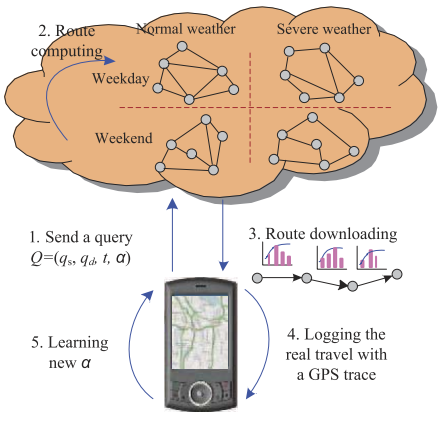


Fig. 8. Framework of self-adapted routing service.

algorithm proposed in [9]. The solution for Principle 2 is similar.

Local smoothing. This step aims to find the longest subsequence from the resulting sequence of the global smoothing so as to satisfy the next-nearest principle. It's clear that the brute-force algorithm which checks all the subsequences (whether satisfy Principle 3) takes exponential time. We propose an polynomial time algorithm as shown in Algorithm 1. Table 1 illustrates a running example when $i = 5$ (see line 1 of Algorithm 1). SL(i), $i = 1, 2, \ldots, 4$ are the sorted lists which store the current longest subsequences beginning with $l_j \rightarrow l_{j_k}, k = 1, 2, \ldots, i$ ordered by $\text{dist}(l_j, l_{j_k})$ ascending, e.g., $\text{SL}(1) = \{l_1 \rightarrow l_3 \rightarrow \cdots, l_1 \rightarrow l_2 \rightarrow \cdots, l_1 \rightarrow l_4 \rightarrow \cdots, \ldots\}$ where $\text{dist}(l_1, l_3) < \text{dist}(l_1, l_2) < \text{dist}(l_1, l_4) \ldots$ For each i, j goes from $i-1$ down to 1, we update the SL(j), e.g., as shown in Table 1b, when $j = 2$, we find $p = 1$ since $\text{dist}(l_2, l_3) < \text{dist}(l_2, l_5) < \text{dist}(l_2, l_4)$. Then, we insert $l_2 \rightarrow l_5$ after the first sequence $\text{SL}(2)^{(1)} = l_2 \rightarrow l_3 \rightarrow l_4$. At line 9 of the algorithm, we lookup the sequence $l_3 \rightarrow l_4 \rightarrow l_5$ from SL(3) but fail to find it, which means this sequence violates Principle 1. Hence, we needn't add l_5 to the end of $\text{SL}(2)^{(1)}$. When $j = 1$, we get $p = 2$ and for $w = 1, 2$, we succeed to find $l_3 \rightarrow l_5$ in SL(3) and $l_2 \rightarrow l_5$ in SL(2), so we add l_5 to both of $l_1 \rightarrow l_3$ and $l_1 \rightarrow l_2$ in SL(1). At the end of Algorithm 1 (line 11), we find the longest sequence among all the SL lists as the result. It's clear that this algorithm takes $O(n^3)$ time (without considering the HASH cost for searching a sequence in SL at line 9). In practise, since the number of landmarks in a rough route is small (usually $n = 10 - 20$ for a 15 km trip), the whole smoothing processing is quite efficient.

Algorithm 1. LocalSmoothing

```
   Input: a sequence L = l_1 → l_2 → l_3 → ... → l_{n-1} → l_n,
          dist(l_i,l_j), i, j = 1, 2, ..., n
   Output: a subsequence (of L)
          L' = l'_1 → l'_2 → l'_3 → ... → l'_{m-1} → l'_m that satisfies:
          ∀i = 1, 2, ... m − 1, dist(l_i, l_{i+1}) = min_{j>i}{dist(l_i, l_j)}
1  for i ← 2 to n do
2      for j ← i − 1 downto 1 do
3          if SL(j) == ∅ then
4              Insert the sequence l_j → l_i to SL(j)
5          else
6              Binary search in SL(j) for the largest integer p such
               that dist(l_j,l_{j_p})≤dist(l_j,l_i), if no such value exists,
               p := 0;
               /* where l_j → l_{j_p} → ··· is the p-th
                  sequence in SL(j)                          */
7              Insert l_j → l_i after the p-th sequence of SL(j);
               /* if p == 0, insert l_j → l_i as the first
                  element of SL(j)                           */
8              for w ← 1 to p do
9                  if SL(j)^(w) ⊖^l l_j ⊕^r l_i ∈ SL(j_w) then
                       /* SL(j)^(w) == l_j → l_{j_w} → ··· is the
                          w-th sequence of SL(j)
                          SL(j)^(w) ⊖^l l_j ⊕^r l_i represents that
                          the SL(j)^(w) removes the first
                          landmark l_j from the
                          beginning(left) and adds l_i to
                          the end(right), i.e.,
                          SL(j)^(w) ⊖^l l_j ⊕^r l_i == l_{j_w} → ··· → l_i
                          */
10                     SL(j)^(w) := SL(j)^(w) ⊕^r l_i;
                       /* add l_i after the sequence
                          SL(j)^(w), i.e.,
                          SL(j)^(w) := l_j → l_{j_w} → ··· → l_i   */
11     return The longest sequence L' in {SL(i)|i = 1, 2, ..., n}
```

4.2 Refined Routing

Suppose after the smoothing, we get a rough route R_{rough}: $q_s \rightarrow l'_1 \rightarrow l'_2 \rightarrow l'_3 \rightarrow \cdots \rightarrow l'_{n-1} \rightarrow l'_n \rightarrow q_d$. This stage finds in the real road network a detailed fastest route that sequentially passes the landmarks of a rough route by dynamic programming. Assume $r_1, r_2, \ldots, r_n$ are the corresponding road segments (Definition 3.1) of $l'_1, l'_2, \ldots, l'_n$, i.e., $r_i = l'_i$. Recall Definition 2.1, each r_i has its start point $r_i.s$ and end point $r_i.e$. Let $f_s(i)$ and $f_e(i)$ be the earliest leaving times (after traversing r_i) at nodes $r_i.s$ and $r_i.e$, respectively. Let $T(a, b, c)$ be the travel time of the fastest route from road nodes a to b without crossing node c. Let $t_{se}(i) = r_i.length/r_i.speed$, i.e., the time (estimated based on speed constraint) for traveling from $r_i.s$ to $r_i.e$, and

$$t_{es}(i) = \begin{cases} t_{se}(i) & \text{if } r_i \textit{ is bidirectional} \\ \infty & \text{if } r_i \textit{ is one} - \textit{way}. \end{cases}$$

Using these notations, we have the initial states $f_s(1)$ and $f_e(1)$ as follows:

$$\begin{aligned} f_s(1) &= T(q_s, r_1.e, r_1.s) + t_{es}(1) \\ f_e(1) &= T(q_s, r_1.s, r_1.e) + t_{se}(1). \end{aligned} \tag{8}$$

As shown in Fig. 6B, let $T^i_{se} = T(r_i.s, r_{i+1}.e, r_{i+1}.s)$ denote the time of the fastest route (using speed constraint in real road network) which starts from point $r_i.s$ and ends at point $r_{i+1}.e$ without crossing $r_{i+1}.s$ in road network G_r. Then, T^i_{ee}, T^i_{ss}, and T^i_{es} can be similarly defined. Now, we have the state transition equations:

$$\begin{aligned} f_s(i+1) &= \min\{f_s(i) + T^i_{se}, f_e(i) + T^i_{ee}\} + t_{es}(i+1) \\ f_e(i+1) &= \min\{f_s(i) + T^i_{ss}, f_e(i) + T^i_{es}\} + t_{se}(i+1). \end{aligned} \tag{9}$$

After $f_s(n)$ and $f_e(n)$ are computed, the total travel time for the optimal route in the real road network is

$$\min\{f_s(n) + T(r_n.s, q_d, r_n.e), f_e(n) + T(r_n.e, q_d, r_n.s)\}.$$

In practise, we can compute T^i_{se}, T^i_{ee}, T^i_{ss}, T^i_{es} and corresponding routes in parallel (for $1 \le i \le n-1$) by utilizing Dijkstra or A*-like Algorithms with a simple modification (by ignoring node c). Then, the final route is a by-product of the dynamic programming since we only need to determine the direction for each landmark road segment.

4.3 Learning Custom Factor

This section describes the process for learning the user's custom factor and providing self-adapted fastest route, which contains five steps as illustrated in Fig. 8:

1. **Query sending**. First, the user send her query tuple (q_s, q_d, t_d, α) to the cloud, where q_s and q_d are start point and destination and t_d is the departure time. The parameter α, is the *custom factor* (Definition 4.1).
2. **Route computing**. According to the departure time, start and destination point, the cloud chooses a proper landmark graph considering the weather information and whether it's a holiday or a workday. Based on the landmark graph, a two-stage routing algorithm is performed to obtain a *time-dependent* fastest route based on Section 4.

TABLE 1
A Running Example of Algorithm 1

(a) SL(j), $j = 1, 2, 3$ before $i = 5$

SL(1)	SL(2)	SL(3)
$l_1 \rightarrow l_3$	$l_2 \rightarrow l_3 \rightarrow l_4$	$l_3 \rightarrow l_4$
$l_1 \rightarrow l_2$	$l_2 \rightarrow l_4$	
$l_1 \rightarrow l_4$		

(b) SL(j), $j = 1, 2, 3, 4$ after $i = 5$

SL(1)	SL(2)	SL(3)	SL(4)
$l_1 \rightarrow l_3 \rightarrow l_5^{(j=1,w=1)}$	$l_2 \rightarrow l_3 \rightarrow l_4$	$l_3 \rightarrow l_5^{(j=3,p=0)}$	$l_4 \rightarrow l_5^{(j=4)}$
$l_1 \rightarrow l_2 \rightarrow l_5^{(j=1,w=2)}$	$l_2 \rightarrow l_5^{(j=2,p=1)}$	$l_3 \rightarrow l_4$	
$l_1 \rightarrow l_5^{(j=1,p=2)}$	$l_2 \rightarrow l_4$		
$l_1 \rightarrow l_4$			

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 for recalculate the user's custom factor. The more a driver uses this system, the deeper this system understands the driver; hence, a better driving direction services can be provided.
5. **Adapting the custom factor**. The custom factor of a given user can be learned in an self-adaptive way. Initially, we assign the user a default value, e.g., 1.0. Let $\alpha^{(M)}$ be the custom factor the client sent to the cloud for the Mth query. Let $CDF_i^{(M)}(\alpha)$ be the cumulative distribution function (refer to Fig. 5b) for the ith landmark edge . After the travel, we calculate the real travel time of this landmark edge $T_i^{(M)}$ by the recorded GPS logs. Then, the mobile client compute the new custom factor by

$$\widetilde{\alpha}^{(M)} = \underset{\alpha}{\operatorname{argmin}} \left(\frac{1}{p} \sum_{i=1}^{p} |\alpha - CDF_i(T_i^{(M)})|^2 \right), \qquad (10)$$

where p is the number of landmark edges. This single-valued minimization problem can be solved using the optimization approaches or just using the simple enumeration method (uniformly trying the α from 0 to 1). To obtain a stable value for α, we need to study the most recent n driving routes of a user instead of a single trip. Meanwhile, near past driving paths should be more valuable in calculating α than those distant past. Therefore, we compute the new personalized α by a weighted moving average [10]

$$\alpha^{(M+1)} = \frac{\sum_{i=1}^{n} i\widetilde{\alpha}^{(M-n+i)}}{\sum_{i=1}^{n} i} = \frac{2}{n(n+1)} \sum_{i=1}^{n} i\widetilde{\alpha}^{(M-n+i)}, \qquad (11)$$

where n is the window length of the moving average. In the next query, the updated $\alpha^{(M+1)}$ will be sent to the cloud.

5 Evaluation

5.1 Settings

5.1.1 Data

Road network. We perform the evaluation based on the road network of Beijing, which consists of 106,579 road nodes and 141,380 road segments.

Taxi trajectories. We build our system based on a real trajectory data set generated by over 33,000 taxis over a period of three months. The total distance of the data set is more than 400 million kilometers and the total number of GPS points reaches 790 million. The average sampling interval of the data set is 3.1 minutes per point and the average distance between two consecutive points is about 600 meters. After the preprocessing, we obtain a trajectory archive containing 4.96 million trajectories.

Real-user trajectories. We use the driving history (ranging from two months to one year) of 30 real drivers recorded by GPS loggers to evaluate travel time estimation. This data are a part of the released GeoLife data set [11], and the average sampling interval is about 10 s. That is, we can easily determine the exact road segments a driver traversed and corresponding travel times.

5.1.2 Framework

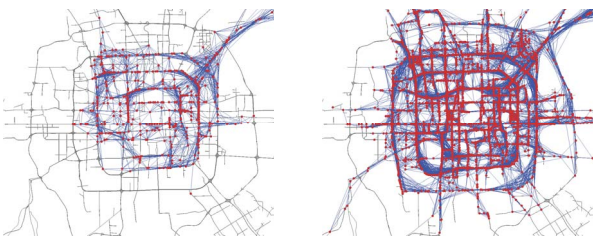
We first validate the capability of our time-dependent landmark graphs in accurately estimating the travel time of a route using user-generated GPS logs. Then, we conduct experiments comparing the routes suggested by different methods using synthetic queries and investigate the effectiveness of the proposed smoothing algorithms. Here, we map a route to a landmark graph and use the travel time estimated by the landmark graph as a ground truth. Finally, rigorous in-the-filed user studies are performed to further explore the performance of our system.

5.2 Evaluation on Travel Time Estimation

5.2.1 Evaluating Landmark Graphs

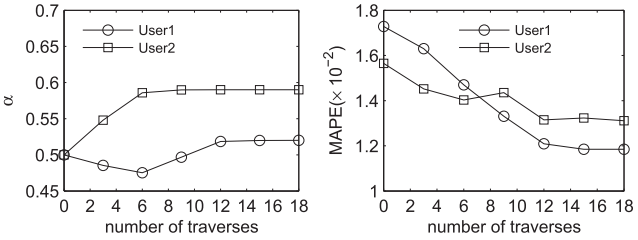
We build a set of landmark graphs with different values of k ranging from 500 to 13,000. The threshold δ is set to 10, i.e., at least 10 times per day traversed by taxis (in total over 900 times in a period of three months) and t_{max} is set to 30 minutes.

Fig. 9 visualizes two landmark graphs when $k = 500$ and $k = 4{,}000$. The red points represent landmarks and blue lines denote landmark edges. Generally, the graph ($k = 4{,}000$) well covers Beijing city, and its distribution follows our commonsense knowledge.



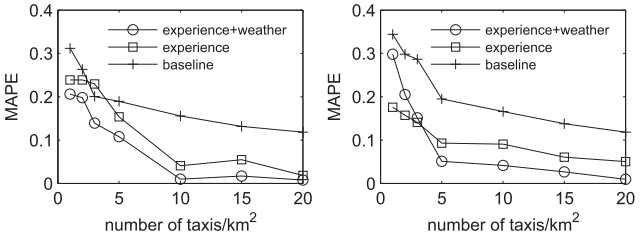
(a) k=500 (b) k=4000

Fig. 9. Visualized landmark graphs.



(a) α learning (b) MAPE

Fig. 10. Custom factor learning.



(a) weekday (b) weekend

Fig. 11. Evaluation on differentiating taxi drivers' experiences.

5.2.2 Learning End Users' Driving Behaviors

We use real users' driving trajectories logged by GPS (released in GeoLife data set [11]) to learn their custom factors using the method proposed in Section 4.3. We measure the accuracy of the travel time estimation using the mean absolute percentage error (MAPE), defined as

$$\text{MAPE} = \frac{1}{N}\sum_i \frac{|t(R_i) - \hat{t}(R_i)|}{t(R_i)}, \tag{12}$$

where $t(R_i)$ is the real travel time of route R_i (obtained from the GPS logs) and $\hat{t}(R_i)$ is the estimated travel time; N is the total number of routes evaluated. Here, we utilize our landmark graph to estimate the travel time of a route by mapping the users' trajectories to the landmark graph (detailed in [1]).

Fig. 10a illustrates the self-tuning process using two users' driving routes recorded in GPS trajectories. Here, their custom factors gradually stabilize after the mobile client processed 10 times of the same route for them, as shown in Fig. 16a). Meanwhile, the error of travel time, measured by MAPE, shows a downward trend with the increasing number of routes processed until reaching 10. Clearly, the two users have different custom factors tuned in different ways. As shown in Fig. 10b, the error measured by MAPE is less than 1.5 percent for both of the users when α becomes stable, which also validates that the landmark graph can well model the dynamic traffic flow and estimate the travel time for a particular user.

5.2.3 Differentiating Taxi Drivers' Experiences

We evaluate the impact of differentiating the taxi drivers' experience (see Section 3.2.2) by investigating the aggregated MAPE of travel time estimation, using the 30 users' GPS trajectories. We study the performance changing over the average number of taxis per km^2 for the following methods: 1) baseline method which does not consider the difference of taxi drivers' experiences; 2) the method differentiating the taxi drivers' knowledge; and 3) the method combing the weather information (using corresponding landmark graph for normal/severe weather) and the difference of taxi drivers' experiences.

As shown in Fig. 11, for both weekdays and weekends, the landmark graph considering the difference of taxi drivers' experiences outperforms the baseline; meanwhile, if the weather information is used, the performance is even higher. With respect to the scale of taxis used for building the landmark graph, the error decreases with the increasing of the scale gradually. Overall, we can get an acceptable performance (MAPE $< 10\%$) as long as there are over eight taxis in a region of 1 km^2.

5.3 Evaluation on Routing

For evaluating the effectiveness of the routes suggested by different methods (say methods A and B), we use the following two criteria: Fast Rate 1 (FR1) and Fast Rate 2 (FR2) where method B is used as a baseline

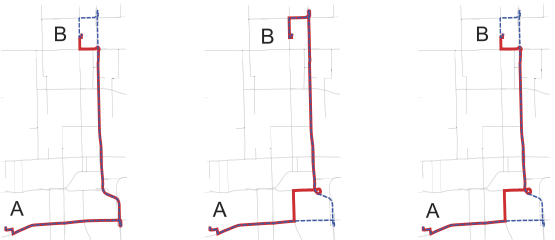
$$\begin{cases} \text{FR1} = \dfrac{\text{Number(A's travel time} < \text{B's travel time)}}{\text{Number(queries)}} \\ \text{FR2} = \dfrac{\text{B's travel time} - \text{A's travel time}}{\text{B's travel time}}. \end{cases} \tag{13}$$

FR1 represents how many routes suggested by method A are faster than that of baseline method B, and FR2 reflects to what extent the routes suggested by A are faster than the baseline's. Meanwhile, we use SR to represent the ratio of method A's routes being equivalent to the baseline's.

5.3.1 Synthetic Origin-Destination Pairs

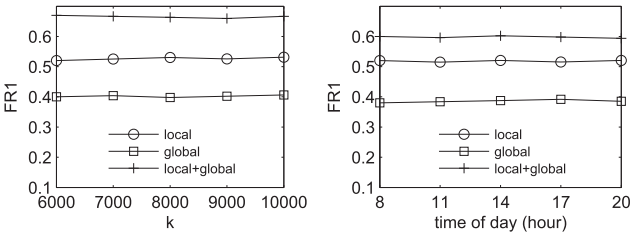
We generate 1,200 queries with different geo-distances of origin-destination pairs and departure times. The geo-distances range from 3 to 23 km and follow a uniform distribution. The departure times range from 6 am to 10 pm and are generated randomly in different time slots.

We first examine whether the proposed smoothing approaches (independently or simultaneously used) can effectively remove the roundabout part of a route and thus reduce the travel time. Fig. 12 visualizes the results for a query (from A to B) at 9 am with a default custom factor (0.5), where the dashed blue line is the baseline route computed by our method without any smoothing. Figs. 12a and 12b present the routes generated by independently applying the local smoothing and global smoothing, respectively. Fig. 12c plots the result combining local and global smoothing. It's clear



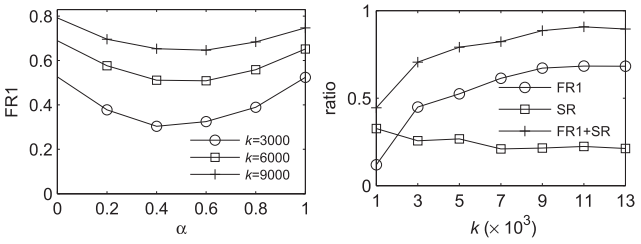
(a) local (b) global (c) local+global

Fig. 12. Visualization of smoothing results.



(a) FR1 w.r.t. # of lanmdarks (b) FR1 w.r.t. time of day

Fig. 13. Overall FR1 of different smoothing approaches, with default $\alpha = 0.5$.



(a) FR1 w.r.t. α (b) FR1/SR w.r.t. k

Fig. 14. Overall performance of T-Drive compared with SC, measured by FR1 and SR.

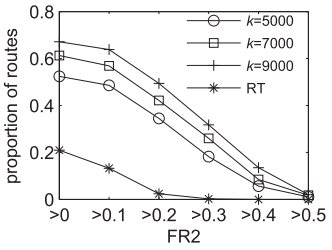


Fig. 15. FR2 over k.

that the proposed smoothing approach removes the roundabout part of the original route. Furthermore, performing the combined smoothing approach is more effective than using global or local smoothing alone.

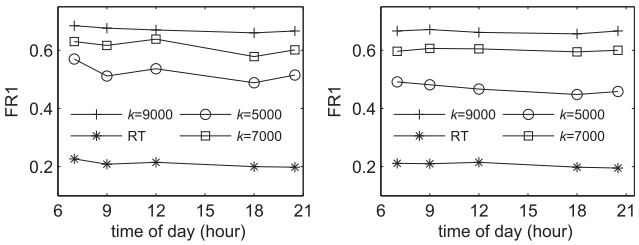
Fig. 13 studies the overall FR1 of the routes induced by different smoothing strategies. Here, we use the two-stage routing approach without the smoothing process as method A in (13), compared with local smoothing, global smoothing as well as the combination of local+global smoothing. We investigate the performance of FR1 with respect to both the number of landmarks (Fig. 13a) and time of day (Fig. 13b). As shown in Fig. 13, more than 60 percent routes suggested by the combined method are better than the baseline (the other 40 percent are the same with the baseline's routes), which significantly outperforms the single local smoothing ($\mathrm{FR1} = 50\%$) and the single global smoothing ($\mathrm{FR1} = 40\%$).

We further compare our approach (combined with the smoothing process) with the speed-constraint-based (denoted as SC) method and a real-time-traffic-analysis-based (termed RT) method in the aspects of efficiency and effectiveness. The SC method (offered by Google[1] and Bing Maps) is based on the shortest path algorithm like A* using the speed constraint of each road segment. The RT method first estimates the speed of each segment at a given time according to the GPS readings of the taxis traversing on the road segment or the road sensor readings [12], and then calculates the fastest route according to the estimated speeds. Note that the "RT" here is not the actually real-time traffic condition, but the estimated speed based on the samples in a near past time interval, e.g., 5 minutes.

1. http://goo.gl/QEvqW, http://goo.gl/7nHi4.

TABLE 2
FR1, SR of TDrive and RT

	α	k	FR1	SR
TDrive	0.4	6,000	0.509	0.281
	0.4	9,000	0.647	0.222
	0.6	6,000	0.511	0.272
	0.6	9,000	0.653	0.216
	0.7	6,000	0.544	0.227
	0.7	9,000	0.672	0.214
RT approach			0.206	0.671



(a) Weekdays (b) Weekends

Fig. 16. FR1 w.r.t. time of day.

Figs. 14 and 15 and Table 2 show the overall performance (FR1, FR2, and SR) of our method. When calculating the FR1, FR2, and SR, both our method and the RT approach use the SC method as a baseline.

Fig. 14 studies the overall FR1 of our method changing over k and α. When $k = 9{,}000$, the lowest FR1 is still over 60 percent, i.e., 60 percent of the routes suggested by our method are faster than that of the SC approach. Fig. 14b further details the FR1 and SR of our method when $\alpha = 0.7$ (due to the page limitation, we only present the results of a few α in the later evaluations). Here, FR1 is being enhanced with the increase of k when $k < 9{,}000$, and becomes stable when $k > 9{,}000$. That is, it is not necessary to keep on expanding the scale of a landmark graph to achieve a better performance. Also, as shown in Table 2, our method outperforms the RT approach in terms of FR1, and most routes (67 percent) suggested by the RT approach are the same as that of the SC method. Fig. 15 plots the FR2 of ours and RT. For example, when $k = 9{,}000$, over 50 percent routes suggested by our method are at least 20 percent faster than the SC approach.

We further study the FR1 of our approach and RT in different time slots. As shown in Fig. 16, both our method and the RT approach have a stable performance in different time slots on weekdays and weekends. Moreover, our method has a 30 percent (on average) improvement over the RT approach when $k \geq 5{,}000$.

The reason why our method outperforms the RT approach is: 1) Coverage: Many road segments have neither embedded road sensors nor taxis traveling on them at a given time. At this moment, the speed constraint of a road segment is used to represent the real-time traffic on the road segment. That is also the reason why the RT approach returns many of the same routes as the SC method. 2) Spareness: Usually, we cannot have enough number of the taxis traveling on a road

TABLE 3
Evaluation Results of the In-the-Field User Study

	Evalution1		Evaluation2	
	Distance	Duration	Distance	Duration
Our System	15.17km	27.15min	15.29km	24.19min
Google	17.24km	31.28min	15.17km	28.63min
Gaps	-2.07km	-4.13min	0.12km	-4.44min
FR1	51.7%	79.4%	35.9%	72.7%
FR2	12.0%	13.2%	-0.791%	15.5%

segment in a near past time interval, e.g., past 5 minutes. Thus, the instant travel time (so called real-time speed) estimated based on these insufficient samples is not very accurate. 3) Open challenges: As compared to the history-based method, the RT approach is more vulnerable to noise, such as traffic lights, human factors (pedestrians crossing a street), and taxis looking for parking places and passengers.

5.3.2 In-the-Field Evaluation

We conduct two types of in the field studies: 1) The *same* driver traverses the routes suggested by our method and a baseline at different times. 2) Two drivers (with similar custom factors learned by our system) travel different routes (recommended by different methods) *simultaneously*.

Table 3 show the results of the two types in-the-field evaluations, where 30 users participated in the Evaluation 1 which last for 10 days and two users are invited to conduct the Evaluation 2 for six days. According to the results, 79.4 percent of the routes provided by our system are better than the baseline with respect to the travel time in the Evaluation 1. On average, we save 15.5 percent time in the Evaluation 2 (T-test: $p < 0.004$) for a 25 min trip.

6 Discussion

To enable our driving direction service in a could environment, some critical issues like efficiency and privacy are investigated.

For revealing the efficiency performance of our method (regardless of the system design), we test our system on a single server with 2.67 GHz CPU and 16 GB RAM (using a single thread without optimization) in the cloud, as shown in Table 4. The mobile client is running on a Windows smartphone with 1 GHz CPU and GPRS connection. Roughly, we can answer 1,000 queries per second using 30 (24-core) servers in a cloud. In the client-side, we only include the items (about 0.1 percent of $|\alpha|$ according to a study) with significant changes, when sending a query to the cloud so as to reduce the transmission cost. In the online phase, the most time-consuming process on the cloud-side is the route computing. The computation cost varies for different road networks in different cities and the size of the landmark graph will change accordingly. Fig. 17b studies the scalability (w.r.t. number of landmarks) of our routing process by using the average number of nodes accessed (when performing routing algorithms in road networks) per query. Obviously, our two-stage routing approach is more efficient than the baselines. According to previous evaluation results (see Fig. 14b), for a large city like Beijing, 9,000 landmarks are enough for our model. Even when k reaches to 12,000, the access cost of our approach is still less than half of the competing methods thanks to the two-stage routing algorithm and parallel routing approach in the refined routing process.

TABLE 4
Time Cost Study on Operations of Our System

Mobile	Time	Cloud	Time
Query sending	144 ms	Rough routing	212 ms
Downloading	287 ms	Refined routing	114 ms
Learning new α	20 us	Building Landmark graph	6 hour

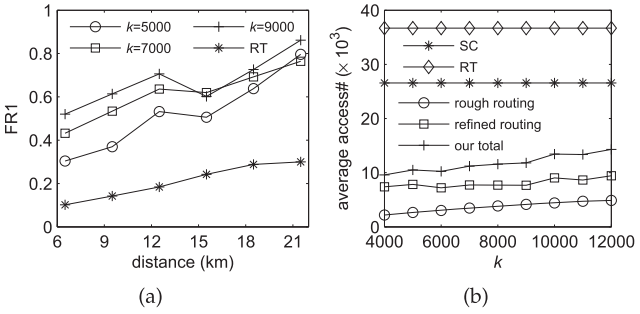


Fig. 17. (a) reveals FR1 of T-Drive and RT method w.r.t. geo-distance of origin-destination pairs; (b) depicts the average number of nodes accessed when performing different routing algorithms in road networks.

As for the privacy issue, the feature of learning the users' driving behaviors can be switched off by the users. Besides, when learning the users' custom factors (Section 4.3), both path recording and α learning are performed in a user's mobile phone. The raw trajectories of the users are not sent to the cloud, only significantly changed custom factors on landmark edges are sent. Therefore, the user's privacy is preserved.

We note for the evaluations based on synthetic queries, though outperforming the baselines, our method still has less than 12 percent (see Table 2, $\alpha = 0.7$, $k = 9{,}000$) of routes falling behind the SC method in terms of FR1. However, after studying these fall-behind routes, we find that they are only slightly (on average, $\mathrm{FR2} = -3\%$, i.e., for a 30 minutes trip, less than 1 minute gap) slower than the SC method. Besides, we use a fixed custom factor in this synthetic-query-based evaluation. However, our system provide the users with personalized routes after learning their driving behaviors. In that situation, the performance of FR1 will be further enhanced. Admittedly, our method is not perfect, since it only leverages the historical data and the challenges mentioned in the Introduction cannot be fully tackled. If real-time sensor data are available for some road segments, our method can be combined to provide better routes for end users. This will be an interesting and challenging work.

7 Related Work

7.1 Driving Direction Services on Web Maps

The shortest or fastest path finding services have been provided by many web maps and local search engines, such as Google, Bing and Yahoo maps, for a long time. Also, most web maps have the function of posting the real-time traffic information on some roads. However, due to the coverage constraints and other open challenges, the real-time traffic condition provided by existing web maps is just

for a user's information while has not been integrated into the driving direction service. In short, the suggest routes are still static (usually calculated based on the distance and speed constraint) and do not vary in time of day.

Our work differs from the existing routing services as follows. First, our driving direction service considers the factor a user, and automatically adapts to the user's driving behavior according to his/her driving paths. Second, we model the historical traffic pattern using the landmark graph, and integrate this information into a time-dependent routing algorithm. Third, we mine drivers' intelligence from taxi trajectories. The intelligence is far beyond the route distance and traffic flows.

7.2 Time-Dependent Fastest Path

The time-dependent fastest path problem is first considered in [13]. Dreyfus [14] suggested a straightforward generalization of Dijkstra algorithm but the authors did not notice it does not work for a non-FIFO network [6]. Under the FIFO assumption, paper [8] provides a generalization of Dijkstra algorithm that can solve the problem with the same time complexity as the static fastest route problem. Demiryurek et al. [15] present a good case study comparing existing approaches for the TDFP problem on real-world networks.

7.3 Traffic-Analysis-Based Approach

There are a few projects [2], [16], [17] aiming to estimate real-time traffic flows and forecast future traffic conditions on some road segments in terms of floating car data [5], [18], [12], such as GPS trajectories as well as Wi-Fi signals. However, these methods are road-segment-level inferences, which predict the traffic conditions on individual road segments with enough samples. As a result, these traffic conditions have not been really applied in the city-wide driving direction services. Recently, Malviya et al. [26] present a system for answering a large number of continuous planning queries in the face of real-time traffic delays with approximation. However, the routes provided to the users are still based on the shortest path without the knowledge from the experienced drivers.

Directly using the inferred real-time traffic condition in a routing algorithm could not find the practically fastest path effectively due to the following reasons: 1) The inferred real-time traffic information could be inaccurate given the insufficient samples from a short time interval. For example, the inferred speed of many service roads and streets (without enough sensors) are not very precise [19]. However, our method using the traffic patterns learned from the long-term historic data is more robust to the sparse data. 2) The essentially needed information for computing the practically fastest path is the traffic condition on a road segment at a future time when the road is actually driven. Using the snapshot of the traffic conditions (on road segments), which maintain the same states of the time when a route is computed, could not be feasible. Instead, our work well models the dynamic city-wide traffic conditions changing over time of day and finds routes by performing a time-dependent routing in the landmark graph.

7.4 History-Learning-Based Approach

Papers [20], [21], [23], [24] present some probabilistic-based methods to predict a user's destination and route based on historical GPS trajectories. Jeung et al. [25] propose a maximum likelihood and greedy algorithm to predict the travel path of an object based on a mobility model. Paper [29] aims to discover popular routes between locations given a huge collection of historical trajectories generated by GPS-enabled devices. Paper [22] computes the fastest route by taking into account the driving and speed patterns learned from historical GPS trajectories.

Our method differs from these methods in the following aspects. First, our goal is to provide users with smart driving directions instead of predicting their path or destinations. Second, we do not explicitly detect speed and driving patterns from the taxi trajectories. Instead, we use the concept of landmarks to summarize the intelligence of taxi drivers. The notion of landmarks follows people's natural thinking patterns, and can improve efficiency of route finding. Third, the routing service considers the driving behaviors of both an end user (for whom the route is being computed) and taxi drivers.

7.5 Driving Directions with Driving Behaviors

Papers like [27], [28] present a few work aiming 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 time-dependent traffic conditions of the dynamic road network learned from experienced taxi drivers and the behavior of the user. Other factors, like day of the week, and weather conditions, are also considered in our routing model.

8 Conclusion

This paper describes a system to find out the practically fastest route for a particular user at a given departure time. Specifically, the system mines the intelligence of experienced drivers from a large number of taxi trajectories and provide the end user with a smart route, which incorporates the physical feature of a route, the time-dependent traffic flow as well as the users' driving behaviors (of both the fleet drivers and of the end user for whom the route is being computed). We build a real system with real-world GPS trajectories generated by over 33,000 taxis in a period of three months, then evaluate the system with extensive experiments and in-the-field evaluations. The results show that our method significantly outperforms the competing methods in the aspects of effectiveness and efficiency in finding the practically fastest routes. Overall, more than 60 percent of our routes are faster than that of the existing online map services, and 50 percent of these routes are at least 20 percent faster than the latter. On average, our method can save about 16 percent of time for a trip, i.e., 5 minutes per 30-minutes driving.

Acknowledgments

This paper is an expanded version of [1], which appeared in Proceedings of ACM SIGSPATIAL 2010 as the Best Paper Runner-up.

REFERENCES

[1] J. Yuan, Y. Zheng, C. Zhang, W. Xie, G. Sun, H. Yan, and X. Xie, "T-Drive: Driving Directions Based on Taxi Trajectories," *Proc. 18th SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems (GIS),* 2010.

[2] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and Travel Time Inference from GPS Probe Vehicle Data," *Proc. Neural Information Processing Systems (NIPS),* 2009.

[3] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-Matching for Low-Sampling-Rate GPS Trajectories," *Proc. Int'l Conf. Advances in Geographic Information Systems (GIS),* 2009.

[4] J. Yuan, Y. Zheng, C. Zhang, and X. Xie, "An Interactive-Voting Based Map Matching Algorithm," *Proc. Int'l Conf. Mobile Data Management (MDM),* 2010.

[5] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding Fastest Paths on a Road Network with Speed Patterns," *Proc. Int'l Conf. Data Eng. (ICDE),* 2006.

[6] A. Orda and R. Rom, "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length," *J. ACM,* vol. 37, no. 3, pp. 607-625, 1990.

[7] N. Leibowitz, B. Baum, G. Enden, and A. Karniel, "The Exponential Learning Equation as a Function of Successful Trials Results in Sigmoid Performance," *J. Math. Psychology,* vol. 54, no. 3, pp. 338-340, 2010.

[8] B.C. Dean, "Continuous-Time Dynamic Shortest Path Algorithms," master's thesis, MIT, 1999.

[9] C. Schensted, "Longest Increasing and Decreasing Subsequences," *Canadian J. Math.,* vol. 13, pp. 179-191, 1961.

[10] W.S. George and G.C. William, *Statistical Methods,* eighth ed. Wiley-Blackwell, 1991.

[11] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web," *Proc. 17th Int'l Conf. World Wide Web (WWW),* 2008.

[12] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic Estimation and Prediction Based on Real Time Floating Car Data," *Proc. 11th Int'l IEEE Conf. Intelligent Transportation Systems (ITSC '08),* pp. 197-203, Oct. 2008.

[13] K. Cooke and E. Halsey, "The Shortest Route through a Network with Time-Dependent Internodal Transit Times," *J. Math. Analysis Applications,* vol. 14, pp. 492-498, 1998.

[14] S. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," *Operations Research,* vol. 17, no. 3, pp. 395-412, 1969.

[15] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "A Case for Time-Dependent Shortest Path Computation in Spatial Networks," *Proc. Int'l Conf. Advances in Geographic Information Systems (GIS),* 2010.

[16] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones," *Proc. Seventh ACM Conf. Embedded Networked Sensor Systems,* 2009.

[17] A. Bejan, R. Gibbens, D. Evans, A. Beresford, J. Bacon, and A. Friday, "Statistical Modelling and Analysis of Sparse Bus Probe Data in Urban Areas," *Proc. Int'l IEEE Conf. Intelligent Transportation Systems (ITS),* 2010.

[18] D. Pfoser, S. Brakatsoulas, P. Brosch, M. Umlauft, N. Tryfona, and G. Tsironis, "Dynamic Travel Time Provision for Road Networks," *Proc. ACM SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems (GIS),* 2008.

[19] A. Gühnemann, R. Schäfer, K. Thiessenhusen, and P. Wagner, "Monitoring Traffic and Emissions by Floating Car Data," *ITS Working Papers,* 2004.

[20] J. Krumm and E. Horvitz, "Predestination: Inferring Destinations from Partial Trajectories," *Proc. Eighth Int'l Conf. Ubiquitous Computing,* pp. 243-260, 2006.

[21] B. Ziebart, A. Maas, A. Dey, and J. Bagnell, "Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior," *Proc. Int'l Conf. Ubiquitous Computing,* 2008.

[22] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag, "Adaptive Fastest Path Computation on a Road Network: A Traffic Mining Approach," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* 2007.

[23] H.A. Karimi and X. Liu, "A Predictive Location Model for Location-Based Services," *Proc. ACM Int'l Conf. Advances in Geographic Information Systems (GIS),* pp. 126-133, 2003.

[24] S.-W. Kim, J.-I. Won, J.-D. Kim, M. Shin, J. Lee, and H. Kim, "Path Prediction of Moving Objects on Road Networks through Analyzing Past Trajectories," *Proc. 11th Int'l Conf. Knowledge-Based and Intelligent Information and Eng. Systems (KES),* pp. 379-389, 2007.

[25] H. Jeung, M.L. Yiu, X. Zhou, and C.S. Jensen, "Path Prediction and Predictive Range Querying in Road Network Databases," *VLDB J.,* vol. 19, pp. 585-602, Aug. 2010.

[26] N. Malviya, S. Madden, and A. Bhattacharya, "A Continuous Query System for Dynamic Route Planning," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 792-803, 2011.

[27] J. Letchner, J. Krumm, and E. Horvitz, "Trip Router with Individualized Preferences (trip): Incorporating Personalization into Route Planning," *Proc. Conf. Innovative Applications of Artificial Intelligence (NCAI),* 2006.

[28] B. Liu, "Route Finding by Using Knowledge about the Road Network," *IEEE Trans. Systems, Man, and Cybernetics, Part A: Systems and Humans,* vol. 27, no. 4, pp. 436-448, July 1997.

[29] Z. Chen, H.T. Shen, and X. Zhou, "Discovering Popular Routes from Trajectories," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 900-911, 2011.

Jing Yuan received the BS degree in computational mathematics from the School of the Gifted Young in 2007. He is currently working toward the PhD degree in the School of Computer Science and Technology, University of Science and Technology of China, under the supervision of Prof. Guoliang Chen, who is an academician of Chinese Academy of Science. Since September 2009, he has been working in the Web Search and Mining Group of Microsoft Research Asia as a full time research intern/visiting student, mentored by Dr. Yu Zheng and Dr. Xing Xie. His research interests include spatial-temporal data mining, time series analysis, and information retrieval. He has published several referred papers in top conferences such as ACM SIGKDD, ACM SIGSPATIAL, Ubicomp, and SSTD. He is a student member of the IEEE and the ACM.

Yu Zheng received the PhD degree in communication and information system from Southwest Jiaotong University in China. He is a researcher at Microsoft Research Asia, which he joined in July 2006. His research interests include location-based services, spatiotemporal data mining, ubiquitous computing, and mobile social applications. Specifically, he loves to explore trajectory data generated by moving objects for location-based social networks, transportation, and urban computing. He is an editorial board member of four international journals, published more 50 referred papers on well-known conferences and journals, such as SIGMOD, SIGKDD, AAAI, WWW, Ubicomp, and *Artificial Intelligence*, and has served more than 30 prestigious international conferences as a chair or program committee member including Ubicomp, IJCAI, WWW, and ACM GIS, etc. So far, he has received three technical transfer awards and 18 patent awards. He is a senior member of the IEEE.

Xing Xie received the BS and PhD degrees in computer science from the University of Science and Technology of China in 1996 and 2001, respectively. He is a lead researcher in the Web Search and Mining Group at Microsoft Research Asia, and a guest PhD advisor for the University of Science and Technology of China. He joined Microsoft Research Asia in July 2001, working on spatial data mining, location-based services, and mobile and pervasive computing. He has served on the organizing and program committees of many international conferences such as WWW, UbiComp, GIS, CIKM, and KDD. He was the program cochair of UbiComp 2011. During the past years, he has published more than 90 referred journal and conference papers. He is also a senior member of both the IEEE and the ACM.

Guangzhong Sun received the BEng degree from the University of Science and Technology of China (USTC) in 2000, and the PhD degree from USTC in 2005. He is currently an associate professor in the Department of Computer Science and Technology, USTC. From July 2006 to December 2007, he worked at the Intel China Research Center (ICRC) as a visiting scholar. From October 2007 to August 2008, he worked at Microsoft Research Asia (MSRA) as a member of faculty development project between MSRA and USTC. His main research interests include web information retrieval, parallel optimization, and combinatorial algorithms. He is a member of the IEEE.