International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.10, pp : 1300-1305 1 Oct 2014

Computing Practically Fast Routes with Cloud-Based Cyber- Physical System

R. Reddy Kishore Reddy, J.Chandra Babu
Annamacharya Institute of Technology & Science, Tirupati ,India.
kishore.it1224@gmail.com ,chandrababu1210@gmail.com

Abstract : ***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 dataset generated by over 33,000 taxis in a period of 3 months, and evaluate the system by conducting both synthetic experiments and in-the-field evaluations. As a result, 60–70% of the routes suggested by our method are faster than the competing methods, and 20% of the routes share the same results. On average, 50% of our routes are at least 20% faster than the competing approaches.***

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

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; 3) A user's driving behaviour. Given the same route, cautious drivers will likely drive relatively slower than those preferring driving very fast and aggressively. Also, users' driving behaviours usually vary in their progressing driving experiences. E.g., travelling 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.

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

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.

In this extension work:

- 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.
- We present smoothing algorithms for removing the roundabout part of the original rough routes.
- We build the improved system by using a real world trajectory dataset generated by 33,000+ taxis in a period of 3 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.

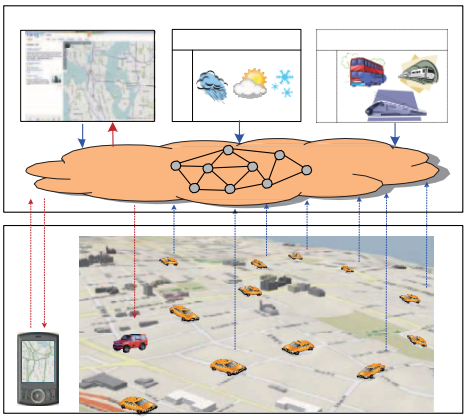

Fig.1: A cloud-based driving directions service

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*=oneway, *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 Gr* is a directed graph, $Gr = (Vr, Er)$, where *Vr* is a set of nodes representing the terminal points of road segments, and *Er* 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., $R : r1 \rightarrow r2 \rightarrow \cdots \rightarrow rn$, where $rk{+}1.s = rk.e$, $(1 \leq k < n)$. The start point and end point of a route can be represented as $R.s = r1.s$ and $R.e = rn.e$. *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., $Tr : p1 \rightarrow p2 \rightarrow \cdots \rightarrow pn$, where $0 < pi{+}1.t - pi.t < \_T$ $(1 \leq i < n)$. $\_T$ 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 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 pre processed 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 travelling 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 "takes 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 *tmax*, two landmarks *u*, *v*, arriving time *ta*, leaving time *tl*, we say $s = (u, v; ta, tl)$ is a *transition* if the following conditions are satisfied:

(I) There exists a trajectory $Tr = (p1, p2, \ldots, pn) \in A$, after map matching, *Tr* is mapped to a road segment sequence $(r1, r2 \ldots, rn)$. $\exists\ i, j,\ 1 \leq i < j \leq n$ s.t. $u = ri$, $v = rj$.

(II) $ri{+}1, ri{+}2, \ldots, rj{-}1$ are not landmarks.

(III) $ta = pi.t$, $tl = pj.t$ and the *travel time* of this transition is $tl - ta \leq tmax$.

Definition 3.3 (Candidate Edge and *Frequency):* Given two landmarks *u*, *v* and the trajectory archive *A*, let *Suv* be the set of the transitions connecting (*u*, *v*). If $Suv \_= \emptyset$, we say $e = (u, v; Tuv)$ is a *candidate edge*, where

$Tuv = \{(ta, tl) | (u, v; ta, tl) \in Suv\}$ 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., $|Suv|$. 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 Gl* = (*Vl*,*El*) is a directed graph that consists of a set of landmarks *Vl* (conditioned by *k*) and a set of landmark edges *E* conditioned by δ and *tmax*. 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 *tmax* 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,2×2 = 4 landmark graphs are built. The weather condition records are crawled from the weather forecast website.

Fig. 3 (A)-(C) 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 (*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 (although the taxi driver only traversed the segment once), as shown in Fig. 3 (C). 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 δ=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. 4(c)) according to the traffic conditions reflected by the raw samples (as shown in Fig. 4(a)) 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. 4(a), the travel times of transitions certaining 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 travelling the landmark edge, and 3) drivers' personal behaviour, 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 pseudo-code), 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 VClustering 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 *Tuv* according to the values of travel time (*tl* − *ta*), and then partition the sorted list *L* into several sub-lists 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

$$WAV(I;L)=\frac{\left|L_A^{(i)}\right|}{|L|}V(L_A^{(i)})+\frac{\left|L_B^{(i)}\right|}{|L|}V(L_B^{(i)}) \qquad (1)$$

where *L*(*i*)*A* and *L*(*i*) *B* are two sub-lists of *L* split at the *i*th element and V represents the variance. This best split point leads to a maximum decrease of

$$\Delta V^{(i)}(L)=V(L)-WAV(I;L) \qquad (2)$$

The algorithm terminates when max*i*{_V(*i*)} is less than a threshold (this will definitely happen due to Theorem 3.6, refer to the appendix part 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, . . . , *cm*}, each of which represents a category of travel times. As shown in Fig. 4(b), 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 $L_B^{(i)}=\{x_j\}_{j=i+1}^{i}$ let

$$\Delta V(i)(L)=V(L)-\frac{\left|L_B^{(i)}\right|V(L_B^{(i)})+\left|L_B^{(i)}\right|V(L_B^{(i)})}{|L|}$$

If$\Delta V(L) = \max i\{\Delta V(i)(L)\}$, then $\Delta V(L)|L| \geq \Delta V(L(i)A)|L(i)A|$ and $\Delta V(L)|L| \geq \Delta V(L(i)B)|L(i)B|$ for $\forall i = 1, 2, \ldots, N$, the equality holds only if $\Delta V(L(i)A) = 0$ and $\Delta V(L(i)B) = 0$ 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 *yi* with the category it certains to ($c(yi)$), and then sort the pair collection $Sxc = \{(xi, c(yi))\}ni = 1$ according to *xi* (arriving time). The information entropy of the collection *Sxc* is given by:

$$Ent(S^{xc}) = -\sum_{i=1}^{m} pi \log(pi) \qquad (3)$$

where *pi* is the proportion of a category *ci* 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 *Sxc* defined as:

$$WAE(i.Sxc) = \frac{|S_1^{xc(i)}|}{Sxc} Ent(S1^{xc}) = -\sum_{i=1}^{m} pi \log(pi)$$

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

$$\Delta E(i) = Ent(S^{xc}) - WAE(i; S^{xc}) - WAE(i; S^{xc})$$

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

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 new-comer. Also, even on the same path, cautious people will likely

Fig. 5. Travel time w.r.t. custom factor 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% 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 learnt travel time distribution. For example, Fig. 5(a) depicts the travel time distribution of an landmark edge in a given time slot ($c1 \sim c5$ denotes 5 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. 5(b). 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 α=0.7, we can find out the corresponding travel time is 272 seconds, while if we set α=0.3 the travel time becomes 197 seconds.

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

"In a network $G = (V,E)$, if A leaves node *u* starting at time $t1$ and B leaves node *u* at time $t2 \geq t1$, then B cannot arrive at *v* before A for any arc (u,v) in *E*". In practise, many networks, particularly transportation networks, exhibit this behaviour [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).

In the rough routing, we first search *m* (in our system, we set *m* = 3) nearest landmarks for *qs* and *qd* 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 the Dijkstra algorithm. For any visited landmark edge, we use the custom factor to determine the travel time. The time costs for travelling from *qs* and *qe* to their nearest landmarks are estimated in terms of speed constraint. For example, in Fig. 6 (A), if we start at time $td = 0$, the fastest route from *qs* to *qd* is $qs \rightarrow r3 \rightarrow r4 \rightarrow qd$. 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 $td = 1$, the route $qs \rightarrow r1 \rightarrow r2 \rightarrow qd$ 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%[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 *qs* to *qd*. 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 $r2 \rightarrow r3$ is very likely to be much longer than the estimated time due to the map matching error, which leads to $r2 \rightarrow r3$ becomes a part of this rough route.

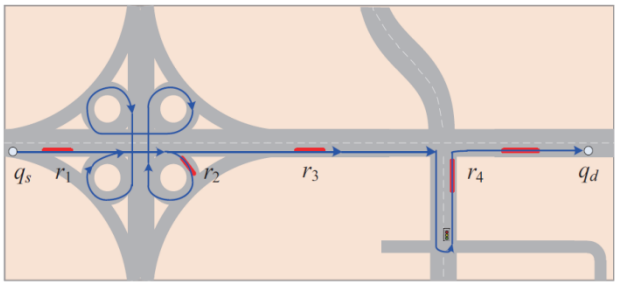

Fig 2 : An example of a roundabout rough route

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 dataset generated by over 33,000 taxis over a period of 3 months. The total distance of the data set is more than 400 million kilometres 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 pre processing, we obtain a trajectory archive containing 4.96 million trajectories.

Real-User Trajectories : We use the driving history (ranging from 2 month to 1 year) of 30 real drivers recorded by GPS loggers to evaluate travel time estimation. This data is a part of the released GeoLife dataset [11], and the average sampling interval is about 10s. 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

Evaluating Landmark Graphs

We build a set of landmark graphs with different values of k ranging from 500 to 13000. The threshold $\delta$is set to 10, i.e., at least ten times per day traversed by taxis (in total over 900 times in a period of 3 months) and *tmax* is set to 30 minutes. Fig. 9 visualizes two landmark graphs when k = 500 and k = 4000. The red points represent landmarks and blue lines denote landmark edges. Generally, the graph (k = 4000) well covers Beijing city, and its distribution follows our commonsense knowledge.

5.3 Evaluation on Routing

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

$$FR1 = \frac{\text{Number(A's travel time<B's travel time)}}{\text{Number(queries)}}$$

$$FR2 = \frac{\text{B's travel time} - \text{A's travel time}}{\text{B's travel time}}$$

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

Synthetic Origin-Destination Pairs*:* We generate 1200 queries with different geo-distances of origin-destination pairs and departure times. The geo-distances range from 3 to 23km and follow a uniform distribution. The departure times range from 6am to 10pm 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 9am with a default custom factor (0.5), where the dashed blue line is the baseline route computed by our method without any smoothing. Fig. 12(a) and Fig. 12(b) present the routes generated by independently applying the local smoothing and global smoothing respectively. Fig. 12(c) plots the result combining local and global smoothing. It's clear 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.

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 Eq. 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 more than 60% routes suggested by the combined method are better than the baseline (the other 40% are the same with the baseline's routes), which significantly outperforms the single local smoothing (FR1=50%) and the single global smoothing (FR1=40%).

6 RELATED WORK

6.1 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. 13 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 behaviour 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.

6.2 Time-Dependent Fastest Path

The time-dependent fastest path (TDFP) problem is first considered in [13]. [14] suggested a straightforward generalization of the 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. [15] presents a good case study comparing existing approaches for the TDFP problem on real-world networks.

7 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 behaviours (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 3 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% of our routes are faster than that of the existing on-line map services, and 50% of these routes are at least 20% faster than the latter. On average, our method can save about 16% of time for a trip, i.e., 5 minutes per 30-minutes driving.

REFERENCES :

i *J. Yuan, Y. Zheng, C. Zhang, W. Xie, G. Sun, H. Yan, and X. Xie, "T-drive: Driving directions based on taxi trajectories," in Proc.GIS. ACM, 2010.*

ii *T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from gps probe vehicle data," in Proc.NIPS, 2009.*

iii *Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in Proc. GIS. ACM, 2009.*

iv *J. Yuan, Y. Zheng, C. Zhang, and X. Xie, "An interactive-voting based map matching algorithm," in Proc. MDM, 2010.*

v *E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in Proc. ICDE, 2006.*

vi *A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length," JACM, vol. 37, no. 3, p. 625, 1990.*

vii *N. Leibowitz, B. Baum, G. Enden, and A. Karniel, "The exponential learning equation as a function of successful trials results in sigmoid performance," Journal of Mathematical Psychology, vol. 54, no. 3, pp. 338–340, 2010.*

viii *B. C. Dean, "Continuous-time dynamic shortest path algorithms," Master's thesis, MIT, 1999.*

ix *C. Schensted, "Longest increasing and decreasing subsequences," Can. J. Math., vol. 13, pp. 179–191, 1961.*

x *W. S. George and G. C. William, Statistical Methods, 8th ed. Wiley-Blackwell, 1991.*

Reddy Kishore Reddy R received the B.Tech Degree in Information Technology from Siddharth Institute of Engineering and Technology, University of JNTUA in 2009.He is currently working towards the Master's Degree in Computer Science, in Annamacharya Institute of Technology and Sciences University of JNTUA.

J.Chandra Babu ,Assistant professor in Computer Science and Engineering at Annamacharya Institute of Tchnology and Sciences