Personalized Route Recommendation using Big Trajectory Data

Jian Dai*†, Bin Yang‡, Chenjuan Guo§, Zhiming Ding†

*University of Chinese Academy of Sciences, Beijing, China
†Institute of Software, Chinese Academy of Sciences, Beijing, China
daijian@nfs.iscas.ac.cn
‡Department of Computer Science, Aalborg University, Denmark
byang@cs.aau.dk
§Department of Computer Science, Aarhus University, Denmark
cguo@cs.au.dk

***Abstract*—When planning routes, drivers usually consider a multitude of different travel costs, e.g., distances, travel times, and fuel consumption. Different drivers may choose different routes between the same source and destination because they may have different driving preferences (e.g., time-efficient driving v.s. fuel-efficient driving). However, existing routing services support little in modeling multiple travel costs and personalization—they usually deliver the same routes that minimize a single travel cost (e.g., the shortest routes or the fastest routes) to all drivers.**

We study the problem of how to recommend personalized routes to individual drivers using big trajectory data. First, we provide techniques capable of modeling and updating different drivers' driving preferences from the drivers' trajectories while considering multiple travel costs. To recommend personalized routes, we provide techniques that enable efficient selection of a subset of trajectories from all trajectories according to a driver's preference and the source, destination, and departure time specified by the driver. Next, we provide techniques that enable the construction of a small graph with appropriate edge weights reflecting how the driver would like to use the edges based on the selected trajectories. Finally, we recommend the shortest route in the small graph as the personalized route to the driver. Empirical studies with a large, real trajectory data set from 52,211 taxis in Beijing offer insight into the design properties of the proposed techniques and suggest that they are efficient and effective.

I. Introduction

Traveling plays an important role in our lives and more and more people choose to use vehicles for traveling. To facilitate route selection, a variety of navigation services become available and are able to recommend routes when a source, a destination, and sometimes, a departure time, are given. However, the routes recommended by existing navigation services are not always preferred by all drivers. For example, a recent study suggests that the routes provided by a leading navigation service often fail to agree with the routes chosen by local drivers [5].

The reason of the disagreement may be two-fold. First, most of the existing navigation services only consider a limited number of travel costs, e.g., distance or travel time, and return routes that minimize a single travel cost, e.g., shortest routes or fastest routes. In contrast, drivers may consider a multitude of different travel costs. For instance, due to an increasing public awareness of environmental protection and high fuel pricing, many drivers increasingly consider fuel consumption [1], in addition to travel times and travel distances.

Second, existing navigation services provide all drivers with the same routes (e.g., shortest routes or fastest routes) and they do not take into account individual drivers' driving preferences (e.g., time-efficient driving, fuel-efficient driving, or some trade-off between them).

These motivate us to study how to model drivers' driving preferences and to provide personalized routes to different drivers, which can better satisfy drivers' needs.

Fig. 1 shows two different drivers's choices of routes from source s to destination d. Both routes have similar distances, however, $route_A$ takes less travel time and $route_B$ takes less fuel[1]. This clearly demonstrates that the two drivers have different driving preferences—one tries to save time and the other aims to save fuel. In many cases, drivers also choose routes according to trade-offs among multiple travel costs of interest. Since different drivers may have different trade-offs, a single, recommended route cannot be preferred by all drivers.

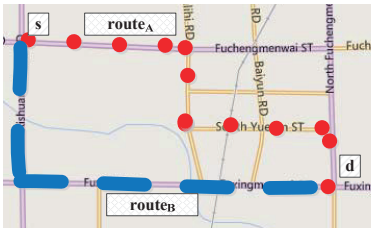


Fig. 1. Routes Used by Two Different Drivers

With the rapid development and continuing use of vehicle tracking technologies (e.g., GPS), big trajectory data becomes available [24], [16]. The big trajectory data provides opportunities to enable better navigation services that consider multiple travel costs and individual drivers' driving preferences. In particular, it is possible to learn and update individual drivers'

[1]Travel time and fuel consumption can be computed based on speed information recorded in GPS trajectories.

driving preferences according to their trajectories. Further, when a driver plans a route, the trajectories used by those drivers who have similar driving preferences to the driver can be utilized to suggest personalized route to the driver.

To the best of our knowledge, this paper is the first to explore the possibility of providing personalized route recommendation using big trajectory data. Specifically, the paper makes four contributions. First, it proposes a novel problem on personalized route recommendation based on big trajectory data. Second, it proposes techniques to model and update driving preferences from drivers' trajectories. Our driving preference model can support arbitrary number of travel costs of interest and distributions of cost ratios. Third, it proposes a local and a global route recommendation algorithms to recommend personalized routes to drivers. The algorithms are novel because (a) reference trajectories are selected from big trajectories while considering driving preferences; (b) local and global route recommendations are proposed to support different routing scenarios. Fourth, it reports on comprehensive experiments conducted on a substantial, real trajectory data set. These elicit design properties of the paper's proposals and characterize the efficiency and the effectiveness of the personalized route recommendation.

The remainder of the paper is structured as follows. Section II defines the driving preference and formalizes the problem. Section III describes the indexes. Section IV describes the retrieval of reference trajectories. Section V presents the personalized route recommendation methods. Section VI reports on the empirical evaluation. Section VII reviews related work and Section VIII concludes the paper.

II. Problem Formulation

A. Basic Concepts

Definition 1: A **road network** is a directed graph $G = (V, E)$, where V is a vertex set and $E \subseteq V \times V$ is an edge set. A vertex $v_i \in V$ denotes a road junction or a road end. An edge $e_k = (v_i, v_j) \in E$ represents a directed road segment, indicating that travel is possible from its starting vertex v_i to its ending vertex v_j. To ease the following discussions, we denote the starting and ending vertices of edge e_k as $e_k.s$ and $e_k.d$, respectively.

Definition 2: A **Route** $\mathcal{R} = \langle r_1, r_2, \cdots, r_A \rangle$ is a sequence of edges, where $r_i \in E$ and $r_i \neq r_j$ if $i \neq j$. The consecutive edges must share a vertex, i.e., $r_i.d = r_{i+1}.s$ where $1 \leqslant i < A$.

Definition 3: A **trajectory** $\mathcal{T} = \langle \langle t_1, p_1 \rangle, \langle t_2, p_2 \rangle, \ldots, \langle t_B, p_B \rangle \rangle$ is a sequence of GPS records pertaining to a trip, where each GPS record $\langle t_i, p_i \rangle$ indicates that a vehicle is at location p_i at timestamp t_i. Further, the GPS records in a trajectory are ordered by their timestamps, i.e., $t_i < t_j$ if $1 \leqslant i < j \leqslant B$. Map matching [17] is used to map a GPS record to a specific location on an edge in the underlying road network. A trajectory $\mathcal{T}$ is also associated with a driver identifier, denoted as $\mathcal{T}.driver$, indicating who made the trajectory.

Definition 4: A trajectory is associated with a **cost vector** $costs(\mathcal{T}) = \langle c_1, c_2, \ldots, c_N \rangle$, where cost value $costs(\mathcal{T}).c_i$ corresponds to the i-th cost of using $\mathcal{T}$.

In this paper, we consider $N = 3$ types of costs—travel distance (i.e., $costs(\mathcal{T}).c_1$), travel time (i.e., $costs(\mathcal{T}).c_2$), and fuel consumption (i.e., $costs(\mathcal{T}).c_3$), which are the three most important factors while planning trips [21]. However, the proposed techniques also apply to cases with arbitrary N.

Given a trajectory $\mathcal{T}$, the travel distance of $\mathcal{T}$ can be obtained by first map-matching the GPS records in $\mathcal{T}$ to road segments and then summing up the lengths of the road segments. The travel time of $\mathcal{T}$ can be obtained by computing the difference between the timestamps of the last and first GPS records in $\mathcal{T}$. The fuel consumption of $\mathcal{T}$ can be derived from the length and the average speed of the trajectory, which can be derived from the GPS records in $\mathcal{T}$, using appropriate vehicular environmental impact models [9].

B. Modeling Driving Preferences

Definition 5: Given two travel costs of trajectory $\mathcal{T}$, say $costs(\mathcal{T}).c_i$ and $costs(\mathcal{T}).c_j$, the **preference ratio** w.r.t. the two travel costs is $pr_{i,j} = \frac{costs(\mathcal{T}).c_i}{costs(\mathcal{T}).c_j}$.

Suppose $costs(\mathcal{T}) = \langle 6.5\ km, 13\ min, 0.58\ l \rangle$, the preference ratio w.r.t. the distance and travel time is $pr_{1,2} = \frac{6.5}{13}$.

Definition 6: A driver is associated with a **driving preference vector** $P = \langle p_1, p_2, \ldots, p_M \rangle$ that consists of M random variables, where $M = \binom{N}{2}$ and each random variable p_i describes the distribution of a preference ratio.

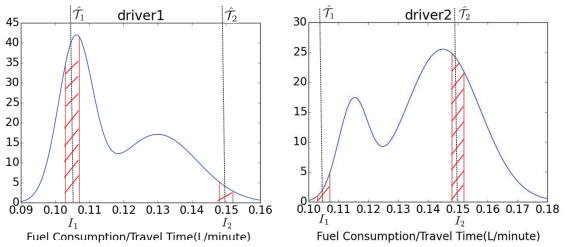
For example, when considering $N = 3$ travel costs, a driver's driving preference vector has $M = \binom{3}{2} = 3$ random variables, which describe the distributions of the preference ratios w.r.t. distance v.s. travel time, distance v.s. fuel consumption, and travel time v.s. fuel consumption, respectively.

Given a driver's historical trajectories, the driver's driving preference vector can be derived as follows. We consider a preference ratio at a time. For each trajectory, the preference ratio of the trajectory can be computed according to Definition 5. Thus, we obtain a collection of preference ratio values from the driver's all trajectories. Next, a random variable describing the distribution of the values in the collection can be obtained. Finally, M random variables corresponding to the distributions of the M preference ratios become available, which in turn construct the driver's driving preference vector.

A Gaussian Mixture Model (GMM) is employed to describe the distribution of a random variable in the driving preference vector P, because a GMM is capable of describing any complex probability functions [3], [21]. Given a collection of values, the GMM can be derived using existing algorithms [19]. Thus, P is actually a sequence of M GMMs.

Definition 7: Given a driver's driving preference vector $P = \langle p_1, p_2, \ldots, p_M \rangle$, a **Personalized Satisfaction Score Function** $\mathcal{F}$ quantifies the degree of satisfaction of a trajectory $\mathcal{T}$ for the driver. In particular, $\mathcal{F}(\mathcal{T}, P) = \sum_{i=0}^{M} \int_{\hat{\mathcal{T}}_i - \Delta}^{\hat{\mathcal{T}}_i + \Delta} p_i(c)dc$, where $\hat{\mathcal{T}}_i$ represents the preference ratio of $\mathcal{T}$ w.r.t. p_i, Δ is a small real value to form a narrow Δ-neighborhood $[\hat{\mathcal{T}}_i - \Delta, \hat{\mathcal{T}}_i + \Delta]$. The higher value returned by function F, the more satisfactory the trajectory is for the driver.

Assume we only consider two travel costs (i.e., fuel consumption and travel time) and thus only one preference ratio in driving preference vector $P = \langle p_1 \rangle$. Fig. 2 shows that the pdfs of the preference ratio random variable p_1 for two different drivers $driver1$ and $driver2$ in our real trajectory data set. Consider two trajectories $\mathcal{T}_1$ and $\mathcal{T}_2$ using two different routes but connecting the same source-destination. We first compute their corresponding preference ratios and construct their Δ-neighborhoods I_1 and I_2, respectively, as shown in Fig. 2. Clearly, for $driver1$, the route used by $\mathcal{T}_1$ is preferred than that by $\mathcal{T}_2$ because $\mathcal{F}(\mathcal{T}_1, P_{driver1}) > \mathcal{F}(\mathcal{T}_2, P_{driver1})$. Similarly, the route used by $\mathcal{T}_2$ is preferred by $driver2$.



(a) pdf of p_1, $driver1$ (b) pdf of p_1, $driver2$

Fig. 2. Evaluating Personalized Satisfaction Score Functions

C. Updating Driving Preferences

A driver's driving preference vector typically does not keep invariant, especially when considering a relatively long time span. For example, when a driver's financial situation changes, e.g., having a high salary job after graduation from university, the driver's driving preference vector may be changed significantly.

To timely update the random variables in a driver's driving preference vector, we try to identify when a driver's driving preference vector starts changing significantly. Once the driver produces a new trajectory $\mathcal{T}$, the trajectory is called significantly different from the driver's driving preference vector P if its personalized satisfaction score is smaller than a threshold ξ, i.e., $\mathcal{F}(P, \mathcal{T}) < \xi$. Here, threshold ξ is introduced to play a role which is similar to the p-$value$ in the conventional statistical significance testing [8]. Once such a significantly different trajectory is identified, we start accumulate the driver's trajectories and build a new driving preference vector P according to the procedure described in Section II-B.

D. Problem Formulation

Definition 8: A **Personalized Route Recommendation (PRR)** query, denoted as $PRR(v_s, v_d, t, P)$, takes as input a source vertex v_s, a destination vertex v_d, a departure time t, and a driver's driving preference vector P. The query returns the shortest route in a *reference graph*. The reference graph is a sub-graph of the road network, where the edges satisfy the following two features: (1) the edges were traversed by the trajectories whose drivers have driving preferences similar to P and that went through from v_s to v_d and occurred at a similar departure time t of a day; (2) an edge in the reference graph is associated with a *weight*, where a lower weight value indicates that more drivers prefer to use the edge.

The construction of reference graph is discussed in Section V.

E. Framework Overview

Fig. 3 gives an overview of the framework for answering personalized route recommendation queries. The framework consists of three major modules, *index construction*, *reference trajectories retrieval*, and *personalized route recommendation*.

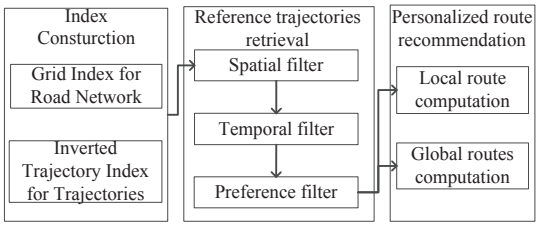


Fig. 3. Framework Overview

The index construction module is responsible for managing the trajectories and the road network where the trajectories occurred. It builds an inverted trajectory index for the trajectories and a grid index for the road network.

Given the input parameters of a PRR query (i.e., a source, a destination, a departure time, and a driving preference vector), the reference trajectories retrieval module applies various filers, including a *spatial filter*, a *temporal filter*, and a *preference filter*, to retrieval a set of *reference trajectories* that are highly relevant to the query. When retrieving the reference trajectories, the two indexes built in the index construction module are used to facilitate the retrieving process.

The personalized route recommendation takes as input the reference trajectories returned by the trajectories retrieval module. Two strategies, *local route computation* and *global route computation*, are applied on the reference trajectories to compute the personalized route satisfying the driving preference vector.

III. Index Construction

A. Indexing the Trajectories

Recall that map matching [17] is able to map a GPS record in a trajectory to a specific location on an edge in the underlying road network. Thus, map matching associates trajectories with edges in the road network.

A common operation in the later reference trajectories retrieval model is to retrieve all the trajectories that occurred on a given edge. To efficiently support this operation, an **Inverted Trajectory Index (ITI)** is constructed to record the relationships between edges and the trajectories occurred on the edges. The ITI contains a set of entries where each entry corresponds to an edge. The entry is in the form of $\langle e, tr\_list \rangle$, where e indicates an edge and tr_list is a *trajectory posting list* that contains the trajectories that occurred on edge e and the trajectories are ordered based on the timestamps of the first GPS records on edge e in the trajectories.

A single scan on the map-matched trajectories is able to construct the ITI. Since the number of trajectories may

be extremely large, it is necessary to design an external memory based indexing construction algorithm, as shown in Algorithm 1.

Algorithm 1: ITI Construction

Input: Trajectories file: $TrajFile$;
Output: ITI file: f;
1 ITI file: $f \leftarrow$new file;
2 int n$\leftarrow$0;
3 **while** *Blocks in TrajFile have not been fully processed* **do**
4 Read next block b from $TrajFile$;
5 Parse the trajectories in block b into $\langle edge, traj\_pointer \rangle$ pairs;
6 Combine the pairs with the same $edge$ into a trajectory posting list tr_list;
7 Build an in-memory ITI index ITI_n and append ITI_n to ITI file f;
8 $n \leftarrow n + 1$;
9 Return f;

Algorithm 1 starts reading in blocks one by one from file $TrajFile$ that contains all the map-matched trajectories (lines 3-4). After loading a block of trajectories, the trajectories are parsed into a set of $\langle edge, traj\_pointer \rangle$ pairs, where $edge$ is an edge that is used by a trajectory and $traj\_pointer$ is a pointer to the trajectory (line 5). Note that the pointer $traj\_pointer$ points to the first GPS record on $edge$ in the trajectory instead of the beginning of the trajectory (i.e., the first GPS record in the trajectory). The pairs with the same edge are grouped together and a trajectory posting list tr_list that contains the edge's $traj\_pointer$ is created (line 6). Thus, an in-memory ITI index for the current block ITI_n can be built. Next, ITI_n is appended to the ITI file f (line 7). Finally, the ITI file f is constructed and returned (line 9).

B. Indexing the Road Network

A uniform grid index is employed to index the edges in the road network. Cells in the grid are squares, and the width of a cell is governed by a parameter c. The grid index splits the entire road network into $x \times y$ cells. The grid index associates each cell with an entry in the form of

$$\langle cell, edge\_set \rangle, \tag{1}$$

where $cell$ indicates a grid cell and $edge\_set$ is a set of edges that are inside the cell. Edge e is inside a cell c_i if its starting vertex $e.s$ or its ending vertex $e.d$ is inside the cell, denoted as $e \in c_i$.

IV. Reference Trajectories Retrieval

Drivers' intelligence on route selection is hidden in the drivers' historical trajectories. To answer a PRR query, it is of interest to identify a set of *reference trajectories* that may be highly relevant to the PRR query to reveal the hidden intelligence.

Given a PRR query, it is neither necessary nor efficient to consider all historical trajectories. For example, when the query concerns the southern part of a city, the trajectories occurred in the northern part of the city may not be very useful. In this section, we define a few *filters* to retrieve the reference trajectories that are highly relevant to the given query.

Assume that we consider a PRR query $PRR(v_s, v_d, t, P)$. We apply the following filters sequentially.

Spatial Filter. If a historical trajectory visited through the starting vertex v_s and then the ending vertex v_d, it may provide important driving intelligence for the query. Thus, such trajectories are considered as reference trajectories. Based on the above, a spatial filter only keeps the trajectories that sequentially visit vertices v_s and v_d and filters out those trajectories that do not.

Given a vertex, it is possible to get a set of edges that start from the vertex or end at the vertex. Let starting edge set $E_s = \{e | e.s = v_s\}$ denote the set of edges that start from vertex v_s and ending edge set $E_d = \{e | e.d = v_d\}$ denote the set of edges that end at vertex v_d.

Recall the definition of the ITI in Section III-A, given an edge, the ITI is able to efficiently return a set of trajectories that occurred on the edge. Let edge $e_s \in E_s$ be an edge from the starting edge set and edge $e_d \in E_d$ be an edge from the ending edge set, respectively. Using the ITI, we have $ITI(e_s)$ and $ITI(e_d)$ that contain the trajectories occurred on edges e_s and e_d, respectively.

Based on $ITI(e_s)$ and $ITI(e_d)$, it is not difficult to identify the trajectories that sequentially visited v_s and v_d. Such a trajectory $\mathcal{T}$ must be in the intersection between $ITI(e_s)$ and $ITI(e_d)$ (i.e., $\mathcal{T} \in ITI(e_s) \cap ITI(e_d)$). Further, the timestamp on v_s should be earlier than that on v_d (i.e., $\mathcal{T}(v_s).t < \mathcal{T}(v_d).t$ where $\mathcal{T}(v).t$ indicates the timestamp when trajectory $\mathcal{T}$ visited vertex v).

Temporal Filter. Since traffic is time-dependent, we consider trajectories that occurred at a similar time of a day to the departure time t of the PRR query. The trajectories occurred closer to the departure time t of the PRR query should provide more accurate driving intelligence than do those trajectories occurred further from t. Hence, we partition a day into m slots and the length of each slot is $\frac{24}{m}$ hours. Further, we also distinguish weekdays from weekends. Let slot I_{PRR} cover the trip starting time t of the PRR query and slot $I_{\mathcal{T}}$ cover the time that trajectory $\mathcal{T}$ visited the source vertex v_s. If a trajectory $\mathcal{T}$ does not satisfy $|I_{PRR} - I_{\mathcal{T}}| \leqslant TFI$, the trajectory is filtered out by the temporal filter, where TFI is a configurable threshold.

Preference Filter. Drivers with different driving preferences may choose quite different routes. Thus, we try to only keep the trajectories that are made by the drivers who have similar driving preference vectors to the preference vector provided by the PRR query. To achieve this, we evaluate the satisfaction score of each trajectory $\mathbb{T}$ that is not filtered yet using the satisfaction score function $\mathcal{F}$. The trajectories with the top-k highest scores are kept.

Note that the preference filter can be skipped if the driving preference vector in the PRR query is not (or can not be) specified, e.g., when a driver uses the PPR service for the first few times.

V. Personalized Route Recommendation

After retrieving the reference trajectories, the personalized route recommendation module recommends routes based on

the reference trajectories. The recommendation process distinguishes two different scenarios based on whether the set of the reference trajectories is empty. If the set is not empty, a local route recommendation algorithm is provided; otherwise, a global route recommendation algorithm is offered.

Both local and global route recommendation follows a generic procedure. First, they construct a reference graph. Next, they assign weights to the constructed reference graph. Finally, they return the route that has the minimum weight as the personalized route for the driver.

A. Local Route Recommendation

When the set of reference trajectories is not empty, we first build a *local reference graph* using the reference trajectories. Next, we assign weights to the local reference graph based on how the graph was used by the reference trajectories. Finally, a shortest path search is conducted on the local reference graph to recommend the personalized routes.

1) Local Reference Graph Construction: A local reference graph $G_{ref} = (V_{ref}, E_{ref})$ is a sub-graph of the road network graph G, where $V_{ref} \subseteq G.V$ and $E_{ref} \subseteq G.E$. The local reference graph is composed of the vertices and edges that are traversed by the reference trajectories. Specifically, if a reference trajectory traversed an edge (v_i, v_j), vertices v_i and v_j are added into V_{ref} and edge (v_i, v_j) is added into E_{ref}.

The local reference graph compactly abstracts the original road network and keeps the most relevant vertices and edges because the reference trajectories consider the spatial, temporal, and preference contexts of the PRR query. Note that a route search on the local reference graph guarantees to find a route from the source vertex to the destination vertex, because the reference trajectory set is not empty.

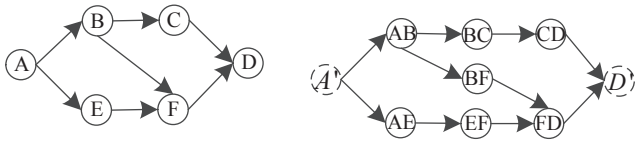
2) Weight Computation: Although the local reference graph only contains the edges that are highly relevant to the PRR query, the edges are not equally important. For instance, an edge used by many reference trajectories is more popular than an edge used by only one reference trajectory, meaning that driver who issued the PRR query should prefer to use the former edge. To this end, we assign each edge a weight reflecting the relative importance of the edge.

Rational of Movement Modeling: The modeling of movements of vehicles on a road network as stochastic processes is well studied in the transportation field [7]. In particular, the modeling of vehicle movements as Markov processes is an easy-to-use and effective approach [22]. Thus, we model the movements of the vehicles in the reference trajectories as a first-order Markov chain. Specifically, we treat each edge as a state. The transition from a state to another state indicates that a movement form an edge to another edge and the corresponding transition probability can be derived based on the reference trajectories. The stationary distribution on each state (i.e., an edge) indicates the probability that a vehicle travels on the edge if the vehicle is driven by a driver with preference $PRR.P$ at time $PRR.t$ from $PRR.v_s$ to $PRR.v_d$. Thus, the stationary distribution value can be used to measure the popularity of the edge.

Since PageRank values on a graph representing the state transition relationship are actually the stationary distribution values [14], we construct a *dual graph* that records the state transition relationship and compute PageRank values on the dual graph.

Dual Graph Construction: A dual graph $\overline{G}_{ref} = (\overline{V}_{ref}, \overline{E}_{ref})$ has a vertex set $\overline{V}_{ref}$ and an edge set $\overline{E}_{ref}$. Each vertex $\overline{v} \in \overline{V}_{ref}$ corresponds to an edge in the local reference graph G_{ref} and each edge $\overline{e} \in \overline{E}_{ref}$ corresponds to a vertex in the local reference graph G_{ref}. To avoid ambiguity, we use the terms *vertex* and *edge* and notation v and e when referring to the local reference graph G_{ref} and use *dual vertex* and *reference edges* and notation $\overline{v}$ and $\overline{e}$ when referring to the daul graph $\overline{G}_{ref}$. A function $\delta : \overline{V}_{ref} \cup \overline{E}_{ref} \rightarrow V_{ref} \cup E_{ref}$ records the relationships between dual vertices and edges and between dual edges and vertices .

Fig. 4(a) shows a local reference graph and Fig. 4(b) shows its dual graph. For example, we have $\delta(AB) = (A, B)$ indicating that dual vertex AB corresponds to edge (A, B), and $\delta((AB, BF)) = B$ indicating that dual edge (AB, BF) corresponds to vertex B. In addition, two additional dual vertices (A' and D') are created in the dual graph to represent the source and the destination, respectively.



(a) Local reference graph (b) Dual graph

Fig. 4. Local reference graph and dual graph

Next, the weights of dual edges in $\overline{G}_{ref}$ is computed based on reference trajectories. Given a dual edge $(\overline{v}_i, \overline{v}_j)$, its weight is computed as

$$\frac{|RefTraj(\overline{v}_i, \overline{v}_j)|}{\sum_{v_x \in OUT(\overline{v}_i)} |RefTraj(\overline{v}_i, \overline{v}_x)|},$$

where $RefTraj(\overline{v}_i, \overline{v}_x)$ returns the set of reference trajectories that traversed edges $\delta(\overline{v}_i)$ and $\delta(\overline{v}_j)$ consecutively, $|\cdot|$ returns the cardinality of a set, and $OUT(\overline{v}_i) = \{\overline{v}_x | (\overline{v}_i, \overline{v}_x) \in \overline{E}_{Ref}\}$. The weight of a dual edge $(\overline{v}_i, \overline{v}_j)$ actually refers to the transition probability from edge $\delta(\overline{v}_i)$ to edge $\delta(\overline{v}_j)$.

After getting the weights of dual edges, the construction of the dual graph completes and classical PageRank computation algorithm [14] can be conducted on the dual graph. In the beginning, the PageRank value of each dual vertex is set to $\frac{1}{|\overline{V}_{ref}|}$. The PageRank values of the dual vertices can be computed according to Equation 2 in an iterative manner until the PageRank values converge.

$$PageRank^{(k)}(\overline{v}_i) = \sum_{\overline{v}_x \in IN(\overline{v}_i)} \frac{PageRank^{(k-1)}(\overline{v}_x)}{OUT(\overline{v}_x)} \quad (2)$$

where $PageRank^{(k)}(\overline{v}_i)$ is the PageRank value of a dual vertex $\overline{v}_i$ in the k-th iteration and $IN(\overline{v}_i) = \{\overline{v}_x | (\overline{v}_x, \overline{v}_i) \in \overline{E}_{Ref}\}$.

The PageRank value of a dual vertex $\overline{v}_i$, i.e., $PageRank(\overline{v}_i)$, actually indicates the probability that a vehicle travels on edge $\delta(\overline{v}_i)$ if the vehicle is driven by a

driver with preference $PRR.P$ at time $PRR.t$ from $PRR.v_s$ to $PRR.v_d$. The higher the PageRank value $PageRank(\overline{v}_i)$ is, the higher probability the edge $\delta(\overline{v}_i)$ may be used by the driver.

To make an edge with a higher PageRank value more likely be chosen in a shortest path finding algorithm, the weight of the edge should be small. To this end, the reciprocal of the PageRank value is employed as the weight of the edge—the weight of edge e is set to $\frac{1}{PageRank(\overline{v}_i)}$ where $\delta(\overline{v}_i) = e$.

3) Local Route Recommendation: Local route recommendation is solved by finding the shortest route on the local reference graph. The shortest route on the local reference graph actually uses the edges that have high chances being used by users with similar driving preference to $PRR.P$, when they travel from $PRR.v_s$ to $PRR.v_d$ at around time $PRR.t$. Algorithm 2 describes the whole procedure of local route recommendation.

Algorithm 2: Local Route Recommendation

Input: ReferenceTrajectorySet: $RTraj$; PRRQuery: PRR;
Output: The Recommended Route: $\mathcal{R}$

1 Construct local reference graph G_{ref} based on $RTraj$;
2 Construct dual graph $\overline{G}_{ref}$ based on G_{ref};
3 Compute the PageRank values of dual vertices in $\overline{G}_{ref}$ according to Equation 2;
4 For each edge $e \in G_{ref}.E_{ref}$, assign value $\frac{1}{PageRank(\overline{v}_i)}$ to edge e if $\delta(\overline{v}_i) = e$;
5 $\mathcal{R} \leftarrow$ the shortest route between $PRR.v_s$ and $PRR.v_d$ on the local reference graph G_{ref};
6 return $\mathcal{R}$;

B. Global Route Recommendation

Global route recommendation considers the scenario where the set of reference trajectories is empty. In this scenario, a set of *transfer edges* which are first identified as intermediate destinations such that non-empty sets of reference trajectories exist between the source and an intermediate destination, between two intermediate destinations, and between an intermediate destination to the destination. Next, the sequence of intermediate destinations that best satisfies the driver's route request is identified and a global reference graph is constructed based on the intermediate destinations. Finally, edge weights are assigned and the shortest route is returned in a way similar to that of local route recommendation.

1) Discover Transfer Edges: Empty reference trajectory set mostly happens when the source and the destination are further away. We propose a *sweep-and-expand* process to discover transfer edges as intermediate destinations. The process considers a *sweep line* that is a directed line segment connecting from the source to the destination. The sweep line indicates the ideal direction that a route should follow from the source to the destination (i.e., a straight line in a Euclidean space). Fig. 5 shows that the sweep line from v_s to v_d.

Next, we decide the directions of the sweep and the expand processes, respectively. We project the sweep line onto the x-axis and y-axis, respectively. If the projected line on the x-axis (or y-axis) is longer, the *sweep process* follows the x-axis (or y-axis), while the *expand process* follows the y-axis (or x-axis). For example, Fig. 5 shows that the x-axis projection of the sweep line is longer. Thus, the sweep process follows the x-axis and is from west to east, because the source vertex v_s is on the west side of the destination vertex v_d. The expand process follows the y-axis and expands to the north and the south simultaneously.

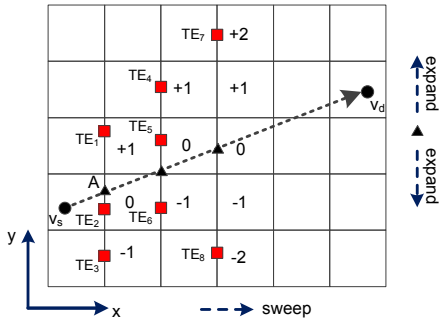


Fig. 5. Discovering Transfer Edges

Recall that a uniform grid index is built to index the edges in the road network. The boundaries of the grid cells must intersect with the sweep line. The intuition of the sweep-and-expand process is that if the route connects from the source to the destination, it must use edges that intersect these boundaries. We choose the edges that intersect the boundaries and are close to the ideal direction of the route as transfer edges. Consider the case shown in Fig. 5. Since the sweep process follows the x-axis, we consider the intersections of the *vertical* cell boundaries and the sweep line. Analogously, when the sweep process follows the y-axis, we should consider the intersections of the *horizontal* cell boundaries and the sweep line.

From each intersection (e.g., A in Fig. 5), we expand to check cells on both directions (i.e., north and south). The cell whose left boundary covers the intersection (the cell with label "0" in Fig. 5), and the cells that are immediately upper (the cell with label "+1" in Fig. 5) and lower (the cell with label "-1" in Fig. 5) than the aforementioned cell are chosen as candidates. The edges that intersect the left vertical boundaries of the candidate cells are chosen as candidate transfer edges.

Given a candidate transfer edge cte), we consider two vertices. The first one is the ending vertex of the candidate transfer edge and the second one is the ending vertex of a candidate transfer edge in the previous sweep step. If this is the first sweep step, the second one is the source vertex v_s. Next, if the set of reference trajectories of the two vertices is non-empty, we keep the candidate transfer edge cte; otherwise, the candidate transfer edge is dropped. If none of the candidate transfer edges is kept in the end, we expand to check more cells and repeat the above procedure. For example, on the 4-th vertical boundary, five cells (i.e., the cells with labels "-2", "-1", "0", "+1", "+2" in the fourth column of cells in Fig. 5) are considered and a transfer edge TE_6 is chosen in cell "+2".

Note that the transfer edge discovering process can be terminated earlier if there exists a non-empty reference trajectory set when considering the ending vertex of a candidate transfer edge and the destination vertex v_d. For instance, if there are a few reference trajectories between the ending vertex of transfer

edge TE_6 and v_d in Fig. 5, then the following two columns of cells can be skipped. We call this strategy *early termination*.

When applying the temporal filter to retrieve reference trajectories in an intermediate step, the departure time from a transfer edge cte needs to be progressively updated. The departure time on a transfer edge is actually an interval. The lower and upper bounds of the interval are the earliest and latest arrival times of the reference trajectories from the previous transfer edge.

2) Finding the Most Likely Transfer Edge Sequence: After obtaining the transfer edges between the source and the destination, we need to identify the most likely sequence of transfer edges. To this end, we transfer the problem into a decoding problem of a hidden Markov model (HMM) [3]. The key is then to construct an appropriate HMM based on the transfer edges. Based on the HMM, a modified Viterbi algorithm is proposed to identify the most likely transfer edge sequence.

Modeling as an HMM: The precondition of using an HMM is to identify a set of states S. Here, we consider each transfer edge as a state. In addition, we also include the source and the destination as states. Thus, we have $S = \{v_s, v_d, TE_1, TE_2, \ldots, TE_6\}$ for the case shown in Fig. 5. The transition probability between two states indicate that the probability of reference trajectories going from one transfer edge or the source to another transfer edge or the destination. The output probability of a state is ignored, because each state can only output the corresponding transfer edge, the source, or the destination itself. Formally, we define an HMM $\lambda = \{\mathcal{S}, \mathcal{M}, \pi\}$, where

- $\mathcal{S}$ is a state set where each state corresponds to a transfer edge, the source, or the destination.
- $\mathcal{M} = \{m_{i,j}\}, 1 \leqslant i, j \leqslant |S|$ is a transition probability matrix, where $m_{i,j} = TP(s_i, s_j)$ is the probability of transferring from state s_i to state s_j.
- $\pi = \{\pi_i\}, 1 \leqslant i \leqslant |S|$ is an initial probability vector. $\pi_i = IP(s_i)$ is the probability of the recommended route starting with the i-th state.

Next, we consider X stages, where the first (or the last) stage corresponds to the start (or the end) of the trip. In other words, in the the first (or the last) stage, only the state that corresponds to the source (or the destination) is allowed. Each of the remaining stages corresponds to a boundary of cells. In such a stage, a set of states that corresponds to the transfer edges intersected with the boundary are allowed. For instance, Fig. 6 shows that there are five stages for the case shown in Fig. 5 and there are 3, 2, and 1 possible transfer edges in the 2nd, 3rd, and 4th stages.

Based on the above, identifying the most likely transfer edge sequence, denoted as $\mathcal{TE} = \langle te_1, te_2, \ldots, te_X \rangle$ (where $te_i \in S, 1 \leqslant i \leqslant X$), is to solve Equation 3.

$$\mathcal{TE} = \arg\max_{\mathcal{TE} \in \mathbb{TE}} P(\mathcal{TE}|\lambda) = \arg\max_{\mathcal{TE} \in \mathbb{TE}} IP(te_1) \prod_{j=2}^{N} TP(te_{j-1}, te_j) \quad (3)$$

where the $\mathbb{TE}$ represents all possible transfer edge sequences.

Deriving π and $\mathcal{M}$: The initial probability of a state is the probability of the recommended route beginning with the state. Thus, if the i-th state corresponds to the source, π_i is set to 1; otherwise, π_i is set to 0.

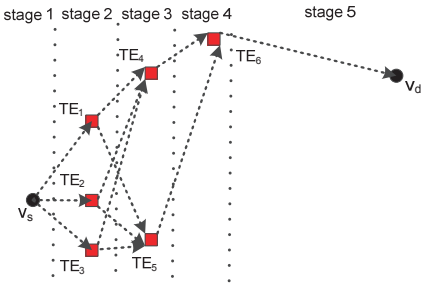


Fig. 6. Selecting the Most Likely Transfer Edge Sequence

The transition probability $m_{i,j} = TP(s_i, s_j)$ is defined in Equation 4.

$$m_{i,j} = \begin{cases} \frac{|RefTraj(s_i, s_j)|}{|RefTraj(s_i)|} & \text{if } s_i \text{ and } s_j \text{ are in two adjacent stages} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

If s_i and s_j are not in two adjacent stages, the transition probability from s_i to s_j is zero. For example, the transition probability from TE_1 in stage 2 to TE_6 in stage 4 is zero. If s_i and s_j are in two adjacent stages, the transition probability equals to the number of reference trajectories that visited both s_i and s_j divided by the number of reference trajectories that visited s_i.

Solving Equation 3: We propose a modified Viterbi algorithm to solve Equation 3. Let $\delta_x(s_i)$ denote the probability of the sequence of transfer edges at stage x is in state s_i, where $1 \leqslant x \leqslant X$ and $1 \leqslant i \leqslant |S|$. We have

$$\delta_1(s_i) = IP(s_i) \quad (5)$$

$$\delta_x(s_i) = \max_{s_j \in S} \delta_{x-1}(s_j) \cdot m_{j,i} \quad (6)$$

Meanwhile, when $2 \leqslant x \leqslant X$, we also record $\phi_x(s_i) = \arg\max_{s_j \in S} \delta_{x-1}(s_j) \cdot m_{j,i}$, i.e., the state in stage $x-1$ that makes $\delta_x(s_i)$ maximized.

In the last stage, i.e., stage X, the state $\arg\max_{s_i \in S} \delta_X(s_i)$ is chosen as the state in the stage X. From this state, the states in the previous stages can be backtracked using $\phi_x(\cdot)$.

In this manner, we can guarantee that the found sequence satisfies the Equation 3 and is optimal. Hence, it always outperforms the other sequences and the benefit of it becomes more significant when planning longer routes.

3) Global Route Recommendation: Solving Equation 3 produces the most likely transfer edge sequence $\mathcal{TE} = \langle te_1, te_2, \ldots, te_X \rangle$, where the first and the last elements te_1 and te_X correspond to the source and the destination, and the remaining elements correspond a transfer edge. Between a pair of elements, say $te_i and te_j$, the reference trajectory set is non-empty, and thus we are able to construct a local reference graph. Finally, the union of all local reference graphs is regarded as the global reference graph. PageRank is conducted on the global reference graph using the corresponding reference trajectories, and edge weights are assigned using the reciprocal of the PageRank values. Finally, the shortest route is recommended as the personalized route to the driver. Algorithm 3 describes the global route recommendation.

Algorithm 3: Global Route Recommendation

Input: PRRQuery: PRR;
Output: The Recommended Route: $\mathcal{R}$
1 Identify transfer edges using the grid index;
2 Build the HMM $\lambda = \{\mathcal{S}, \mathcal{M}, \pi\}$;
3 Identify the most likely transfer edge sequence $\mathcal{TE}$;
4 Construct the global reference graph and assign edge weights;
5 $\mathcal{R} \leftarrow$ the shortest route between $PRR.v_s$ and $PRR.v_d$ on the global reference graph; **return** $\mathcal{R}$;

VI. Empirical Studies

A. Experimental Setup

GPS records: We use more than 50 billion GPS records collected from 52,211 taxis in Beijing, during 2012-09-30 to 2012-11-30. The sampling rate of the GPS records is at least 0.2 Hz—one GPS record is collected in every no more than 5 seconds. A map-matching algorithm [17] is employed to determine the route used by a trajectory[2].

Road network: We consider the road network of Beijing within the 6-th ring road because most of the GPS records are collected within the 6-th ring road. The road network is with 28,342 vertices and 38,577 edges and can be bounded by a 60 km×60 km square region.

Trajectories: We treat each *taxi trip* as a trajectory, where a taxi trip starts when a passenger got in the taxi and ends when the passenger got off the taxi. The information of when a passenger got in and got off a taxi is also recorded in the GPS records. After that, we get 32,379,248 trajectories in total. We select 25%, 50%, 75%, and 100% of the all trajectories to construct four trajectory sets with varying sizes, denoted as TR_1, TR_2, TR_3, and TR_4.

Travel Costs: We consider three commonly used travel costs—travel distance (*TD*) travel time (*TT*) and fuel consumption (*FC*). The travel distance of a trajectory is the sum of the lengths of the edges in the trajectory. The lengths of all edges are recorded in the road network. The travel time of a trajectory is obtained as the difference between the corresponding time points of the last and first GPS records of the trajectory. We use the SIDRA-running model [4] to estimate the fuel consumption based on the length and the average speed of a trajectory. A recent benchmark [9] indicates that SIDRA-running is appropriate for this purpose.

Parameters: We vary important parameters used in the paper. Table I lists these parameters where the default values are shown in bold. Default values are used unless stated otherwise. Parameter TFI is the threshold used in the temporal filter (in Section IV), parameter ξ is the threshold used when updating the driving preference vectors (in Section II-C), and parameter c denotes the width of a cell in the grid index (in Section III). Parameters ED_{LR} and ED_{GR} denote the Euclidean distance between the source and the destination for local route recommendation and global route recommendation, respectively. Parameter TR represents different historical trajectory data sets.

TABLE I. Parameter Settings

Parameters	Values
TFI	**15**, 20, 25, 30 (minutes)
c	1, 2, 3, 4, **5**, 6 (km)
ξ	0.1,0.2,0.3,**0.4**
ED_{LR}	2, 2.5, **3**, 3.5, 4 (km)
ED_{GR}	**10**, 20, 40, 60 (km)
TR	TR_1, $\boldsymbol{TR_2}$, TR_3, TR_4

Other used parameters are described as follows. Parameter Δ used in evaluating personalized satisfaction score function in Definition 7 is set to $\frac{|r_{max}-r_{min}|}{50}$, where r_{max} and r_{min} refer to the maximum and minimum ratios obtained from historical trajectories. Top-5 trajectories are selected by the preference filter, i.e., setting k to 5.

Implementation Details: All algorithms are implemented in Python 2.7.6. A computer with Windows 8 Professional operating system, a 2.5GHZ i5-3210M CPU, and 16 GB main memory is used for all experiments.

B. Driving Preference Modeling and Update

To test the effectiveness of proposed driving preference modeling and update methods, we randomly choose 500 taxi drivers. Each of the driver has at least 500 trajectories. For each driver, we initialize a driving preference vector using the driver's trajectories that are generated in the first month, i.e., 2012-09-30 to 2012-10-31.

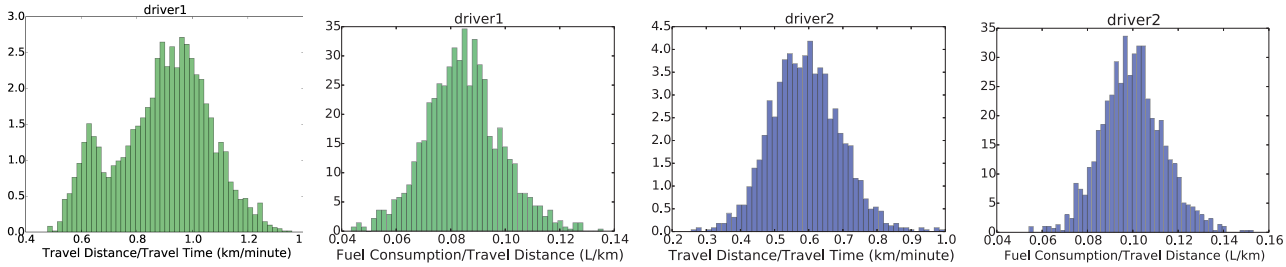
Since we consider three travel costs, i.e., *TD*, *TT*, and *FC*, the driving preference vector $P = \langle p_1, p_2, p_3 \rangle$ is composed of three random variables, where random variables p_1, p_2, and p_3 indicate the distributions of the preference ratios w.r.t. $\frac{TD}{TT}$, $\frac{FC}{TD}$, and $\frac{FC}{TT}$, respectively.

Fig. 7 shows two drivers' driving preferences. In particular, Figs. 7(a) and 7(b) show the distributions of random variables p_1 and p_2 of the first driver, and Figs. 7(c) and 7(d) show the distributions of random variables p_1 and p_2 of the second driver. The distributions of p_3 for both drivers are omitted due to the space limitation. Comparing Fig. 7(a) and Fig. 7(c), it suggests that the second driver drive more aggressively (i.e. the average speed is relatively high) than the first driver.

To evaluate the accuracy of the obtained driving preference vectors, we split a driver's trajectories into training trajectories and testing trajectories. Specifically, we sort a driver's trajectories according to when the trajectories occurred. The training trajectory sets contain the driver's first 20%, 30%, 40%, and 50% of the trajectories, respectively; and the remaining trajectories are the testing trajectories. We compute two preference vectors P_{train} and P_{test} using the training and testing trajectories, respectively. The differences between two preference vectors (P_{train} and P_{test}) are computed by $KLDist = \frac{1}{M} \cdot \sum_{i=1}^{M} KL(GMM_{p_i \in P_{train}} || GMM_{p_i \in P_{test}})$, where $KL(GMM_1 || GMM_2)$ indicates the $Kullback\text{-}Leibler$ divergence [13] between two GMMs. The smaller the $KLDist$ value between the two preference vectors is, the more accurate the derived driving preference vector is.

Fig. 8(a) shows the correlation between the $KLDist$ values and the sizes of the training trajectories, on all driver and

[2]Since the speed limit of most roads in Beijing within the 6-th ring road is 70 km/h, the distances between consecutive GPS records are typically within $5 \cdot \frac{70 \cdot 10^3}{3600} \approx 97$ meters. The average length of a road segment in Beijing is around 200 meters, which indicates that there typically exists at least one or two GPS records per road segment.

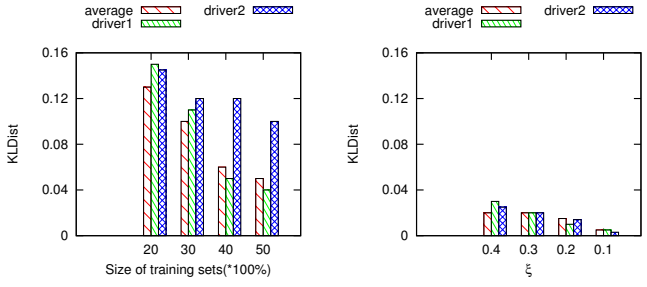


(a) p_1, $\frac{TD}{TT}$, driver1 (b) p_2, $\frac{FC}{TD}$, driver1 (c) p_1, $\frac{TD}{TT}$, driver2 (d) p_2, $\frac{FC}{TD}$, driver2

Fig. 7. Driving Preference Modelling

two specific drivers, *driver1* and *driver2*. On average, the more training trajectories are considered, the lower $KLDist$ values are obtained, which suggests that the accuracy of the driving preference modeling increases as the number of training trajectories increases. See the cases with "all" and "*driver1*" in Fig. 8(a). This can be regarded as a naive driving preference update strategy that driving preference vectors are updated when new trajectories are accumulated. Please note that Fig. 8(a) doesn't use the update strategy thus ξ is not specified. It shows that simply increasing the number of training trajectories cannot always effectively improve $KLDist$.

However, Fig. 8(a) also suggests that the accuracy of *driver2*'s driving preference vector cannot be improved significantly as the number of the driver's trajectories increase. This may be due to the fact that *driver2*'s driving preference changes significantly in the middle. Hence, it is necessary to detect when the *driver2*'s driving preference starts changing significantly, and re-initialize a driving preference vector according to the strategy described in Section II-C. Fig. 8(b) shows the relationship between the parameter ξ and the $KLDist$ values. After updating the GMM, the $KLDist$ values significantly decrease, indicating that the proposed driving preference update strategy is much more effective than the naive update strategy. A smaller ξ can lead to smaller error values, meaning that we are able to successfully detect small driving preference changes. But a small ξ also means that we may need to update the driving preference frequently and is at the risk of over-fitting.



(a) Varying Sizes of Training Sets (b) Varying ξ

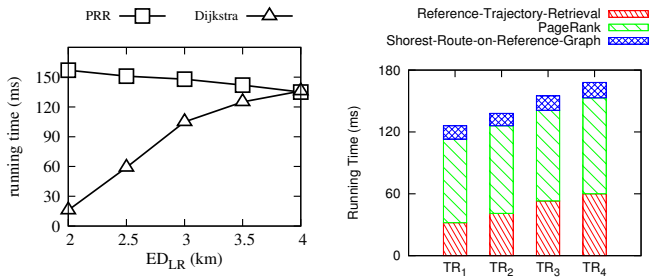
Fig. 8. Driving Preference Update

Note that the driving preference vector P is an input parameter of a PRR query, which is only loosely coupled with the PRR query processing methods. That is, if other algorithms that are able to model and update the driving preference vectors more accurately than do the algorithms proposed in Sections II-B and II-C, they can be applied directly without changing the PRR query processing methods.

C. Efficiency Study

Local Route Recommendation: We study the efficiency of processing a PRR query that only consists local route recommendation. We choose 5 groups of source-destination pairs. In each group, we choose 50 source-destination pairs and the Euclidean distances between the sources and the destinations are with a certain distance ED_{LR} (i.e., 2 km, 2.5 km, 3 km, 3.5 km, 4 km, respectively, according to Table I). For each group, we generate a PRR query for each source-destination pair, with a randomly generated departure time and a randomly chosen taxi driver's preference vector.

Figure 9(a) reports the average run-time of local route recommendation in different settings. The average run-time decreases as the distance between the source and the destination increases. This seems counter-intuitive but it is because we only consider the local reference graph when conducting local route recommendation. And a local reference graph is much smaller compared to the original road network. When the source and the destination are further away, the corresponding reference trajectory set is small which also makes a small local reference graph. PageRank computation and shortest route finding on a small local reference graph is very fast. On average, local route recommendation takes less than 180 ms, which suggests that it is efficient enough to be able to provide real-time local route recommendation services.



(a) Varying ED_{LR} (b) Varying TR

Fig. 9. Local Route Recommendation

To provide a better understanding on how efficient the local route recommendation is, we also report on the average

run-time of identifying the shortest route using the Dijkstra's algorithm in Figure 9(a). The Dijkstra's algorithm takes longer run-time as the distance between the source and the destination increases, because more vertices and edges in the road network need to be explored. Please note that in the last setting, i.e., when the source and the destination are located around 4 km, the Dijkstra's algorithm takes even longer average run-time than does the local route recommendation.

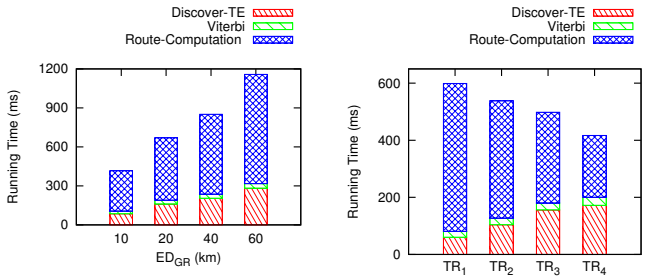
To further investigate how the run-time of the local route recommendation distributes, we report average run-time on each step in Figure 9(b). The PageRank computation step costs the most run-time since it requires many iterations to converge. The route computation time is small because it is conducted on a small local reference graph.

We also vary the sizes of the trajectory data sets. As the number of trajectory grows (form TR_1 to TR_4), the run-time of the local route computation only slowly increases (from 129 ms to 176 ms). This is because the number of reference trajectories grows as more trajectories are considered, thus resulting a larger local reference graph. Nevertheless, when the biggest trajectory set TR_4 with all trajectories is used, the local route recommendation can still finish within a short time (i.e., 176 ms on average), which suggests that local route recommendation can be answered in interactive time.

Global Route Recommendation: Global route Recommendation is invoked when there are no reference trajectory between the source and the destination. This mainly happens when the source and the destination are further away. Thus, in this set of experiments, we choose source-destination pairs that are at least 10 km away from each other. In particular, we choose 4 groups of source-destination pairs. In each group, We choose 50 source-destination pairs and the Euclidean distances between the sources and the destinations are with a certain distance ED_{GR} (i.e., 10 km, 20 km, 40 km, 60 km according to Table I). Note that 60 km is a fairly long distance inside a city, even in a metropolis like Beijing. Similar to the local route recommendation, for each group, we generate a PRR query for each source-destination pair, with a randomly generated departure time and a randomly chosen taxi driver's preference.

Figure 10(a) shows the run-time while varying the distance between the source and the destination. It suggests that (1) the route computation consisting of the three common steps takes the most considerable amount of time, especially the PageRank step; (2) discovering transfer edges and deciding the most likely transfer edge sequence are quite efficient; (3) global route recommendation is scalable as the transfer edge sequence normally contains no more than ten transfer edges in a metropolis due to "early termination". When ED_{GR} is 60 km, the runtime is less than 1.2 seconds, which is sufficient for on-line use.

Next, we evaluate the effect of the sizes of trajectory sets. Counter-intuitively, the average global route computation run-time decreases when the number of trajectories grows, as shown in Figure 10(b). The reason is as follows. Due to the capability of early termination offered by the transfer edge discovery procedure, the global route computation can often be conducted with examining only the first few cells that are close to the source in the grid index, especially when the number of historical trajectories is large enough. Thus, the number



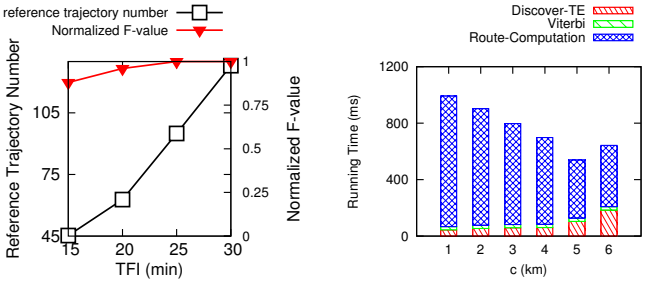
(a) Varying ED_{GR} (b) Varying TR

Fig. 10. Global Route Recommendation

of involved edges is greatly reduced. Accordingly, the overall global route computation run-time decreases as the number of historical trajectories grows. Please notice that the route-computation contains the three steps same as the local route computation.

To summarize, a PRR query, no matter it involves local or global route recommendation, can be returned within 1 second. This suggests that the proposed PRR query processing method is efficient and is able to provide online service.

Effects of TFI and c: Fig. 11(a) depicts the effect of threshold TFI used in the temporal filters. Using a larger TFI significantly increases the number of reference trajectories. However, the increased number of reference trajectories does not significantly increase the $\mathcal{F}$-value (they are all above 0.85). Further, 15-minute is often regarded as the minimal period that traffic can change substantially in the transportation research area [20]. Hence, we use $TFI = 15$ minutes as default settings.



(a) Varying *TFI* (b) Varying c

Fig. 11. Varying Parameters

We also vary parameter c that is the width of a grid cell and report the results in Fig. 11(b). The run-time decreases when c increases from 1 km to 4 km. Later, the run-time starts increasing again when c is greater than 5 km. The reason is two-fold. First, a smaller cell often leads to more sweep-and-expand processes included in the transfer edge discovery phase. Accordingly, more local route recommendations are invoked which take more time. Second, more edges need to be loaded from a larger cell when checking whether they are intersected with cell boundaries, which is also time-consuming. Fig. 11(b) suggests that setting c to 5 km is the optimal setting for our data set because the global route recommendation has the least average run-time. Thus, we choose 5 km as the default setting as shown in Table I.

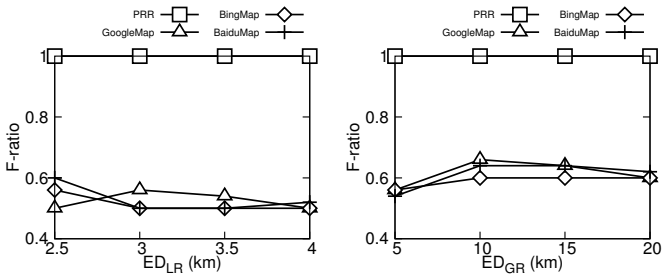
D. Effectiveness Study

To study the effectiveness of the PRR queries, we compare the route recommended by PRR queries with the routes offered by three well-known commercial online map services—Google Map[3], Bing Map[4], and Baidu Map[5].

We first report two case studies using local and global route recommendation respectively, as shown in Fig. 12. Fig. 12(a) shows a case for local route recommendation. Route_1 (which is the shortest path) is returned by Bing map and Google map and route_3 is offered by Baidu map. Note that they recommend a single route to all drivers. Different from these commercial route recommendation services, PRR is able to recommend various routes to individual drivers according to their driving preference. If a driver prefers to perform eco-driving (e.g., driver2 shown in the Fig. 7), route_1 is recommended because it costs less fuel. If a driver prefers to travel with high speeds (e.g., driver1 shown in the Fig. 7), route_2 is recommended.

Fig. 12(b) shows a case for global route recommendation. Route_1 is recommended to a driver who mainly cares about the fuel consumption (i.e., whose ratio $\frac{FC}{TD}$ is small). Fortunately, there are sufficient historical trajectories to support recommending route_1. Hence, route_1 is returned using local route recommendation. Route_2 is returned by global route recommendation using three transfer edges.

To quantify the satisfaction degrees of two different routes, we introduce the $\mathcal{F}$-ratio$= \frac{\mathcal{F}(P,\mathcal{R}')}{\mathcal{F}(P,\mathcal{R})}$. Route $\mathcal{R}$ is recommended by an PRR query and route $\mathcal{R}'$ represents a route returned by an on-line map service or an PRR query. Since some computed routes are pieced up by multiple historical trajectories, we estimate the corresponding costs of them by considering fine-grained costs associated with the involved road segments, which may introduce some error. We plot $\mathcal{F}$-ratios for local route recommendation and global route recommendation in Fig. 13(a) and Fig. 13(b), respectively. Although the three on-line map services may employ different algorithms to return different routes, their average $\mathcal{F}$-ratios exhibit almost the same behavior. Clearly, our proposal outperforms all the three on-line map services, because none of the routes return by the three on-line map services have $\mathcal{F}$-ratios exceeding 1.



(a) Local Route Recommendation (b) Global Route Recommendation

Fig. 13. Effectiveness Study using $\mathcal{F}$-Ratios

Specifically, for local route recommendation, the routes returned by the on-line map services have $\mathcal{F}$-ratios lower than 0.6 as shown in Fig. 13(a); and for global route recommendation, they have $\mathcal{F}$-ratios lower than 0.7 as shown in Fig. 13(b). These demonstrate that our PRR queries are very effective. We notice that the routes returned by the on-line map services have higher $\mathcal{F}$-ratios in global route recommendation than that in local route recommendation. This is because people tend to use express ways and not many options can be provided.

[3]http://ditu.google.cn/

[4]http://cn.bing.com/maps/

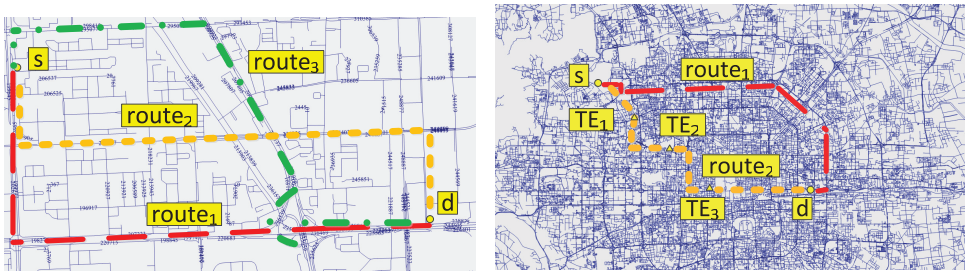
[5]http://map.baidu.com/

VII. Related Work

Route Planning: Various route planning algorithms have been developed to support different scenarios. *Shortest path planning* is supported by Dijkstra's algorithm and A* algorithm, where the edge weights correspond to the travel distances of the edges. *Fastest route planning* relies on the modeling of edges' travel times. A routing algorithm is able to find the fastest paths when modeling the travel time on an edge as a linear function [11]. Another fastest routing algorithm supports the modeling of uncertain travel times, where the travel time on an edge is modeled as a random variable [10]. *Eco-routing* aims to return the route that is the most eco-friendly [1], where the edges are associated with eco-weights reflecting the fuel consumption of traversing the edges. Various methods are proposed to assign such eco-weights [19], [22] using GPS data and conduct eco-routing on the obtained eco-weights [21]. *Skyline routing* considers more than one travel cost, such as distances, travel times, toll fees, and the number of traffic lights, etc., when planning routes. Various algorithms are proposed to support Skyline routing when the edge weights are deterministic [12] and uncertain [21], respectively. All the aforementioned proposals do not explore how to employ drivers' past trajectories to improve the quality of route planning. Further, they do not consider driver's distinct driving preferences and are only able to suggest the same routes to all drivers. In contrast, our proposals utilize drivers' past trajectories to recommend personalized routes to individual drivers.

Route Planning Using Trajectories: Recently, a few studies employ historical trajectories to improve routing services. Given a source and a destination, the most popular route (MPR) between them is identified from historical trajectories [6]. MPR is typically not the shortest route between them and is arguably better than the shortest route. Time period-based most popular route (TPMPR) improves MPR by considering the temporal context. Given a time period, TPMPR is able to identify the MPR during the given time period. Another study focuses on finding top-k popular routes (TKPR) from uncertainty trajectories generated by low frequency GPS data. However, all these proposals do not consider drivers' driving preferences. Namely, they return the same MPR, TPMPR, or TKPR to all drivers.

Personalized Route Planning: Our work significantly differs from TRIP [15]. First, we use big trajectory data from all drivers (using filters to identify relevant ones) but TRIP only uses drivers' own trajectories. Thus, TRIP only works on roads travelled by the driver himself, while ours works on all roads covered by big trajectory data. Second, our empirical study is conducted on a much larger data set while TRIP's is from 100 cars within 2 weeks. Third, TRIP can only model preference w.r.t. travel time, while ours can model arbitrary number of travel costs. Four, we use distributions to model



(a) Local Route Recommendation (b) Global Route Recommendation

Fig. 12. Case Studies

driving preferences rather than average values used by TRIP, which is more accurate.

T-drive [23] is able to return different fastest routes to different drivers. However, the T-driver approach cannot solve our problem because T-drive models a driver's driving preference while considering only one travel cost, i.e., travel time. In contrast, we model a driver's driving preference while considering multiple travel costs.

In a recent work [2], authors assume drivers know all the pareto-optimal routes, but we do not make such unrealistic assumptions. Furthermore, [2] can only model preferences on static travel costs (e.g., distance, number of traffic lights), while we can model dynamic costs (e.g., travel time and fuel) depending on e.g., individual drivers or time of a day.

An interactive personalized route planning system [18] is described. However, it requires users to provide training data, e.g., routes preferred by drivers and routers not preferred by drivers, to identify driving preferences. In contrast, our proposals do not require users to provide any training data, but just their historical trajectories.

VIII. Conclusion and Outlook

We propose and study personalized route recommendation problem using big trajectory data. We provide techniques for modeling and updating driver's driving preferences. We also provide efficient and effective methods to recommend personalized routes in two different settings: local route recommendation and global route recommendation. Empirical studies with a large, real trajectory data set suggest that the paper's proposals are efficient and effective.

In future work, it is of interest to include resources with richer semantic (e.g., points-of-interest) to perform personalized route recommendation. It is also of interest to integrate other driving preference models into the personalized route recommendation framework.

References

[1] O. Andersen, C. S. Jensen, K. Torp, and B. Yang. Ecotour: Reducing the environmental footprint of vehicles using eco-routes. In *MDM (1)*, pages 338–340, 2013.

[2] A. Balteanu, G. Jossé, and M. Schubert. Mining driving preferences in multi-cost networks. In *Advances in Spatial and Temporal Databases*, pages 74–91. Springer, 2013.

[3] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[4] D. Bowyer, R. Akcelik, and D. Biggs. Guide to fuel consumption analysis for urban traffic management. Technical report, ARRB Internal Report-AIR 390-9, 1984.

[5] V. Ceikute and C. S. Jensen. Routing service quality - local driver behavior versus routing services. In *MDM (1)*, pages 97–106, 2013.

[6] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *ICDE*, pages 900–911. IEEE, 2011.

[7] C. F. Daganzo and Y. Sheffi. On stochastic models of traffic assignment. *Transportation Science*, 11(3):253–274, 1977.

[8] R. A. Fisher. The design of experiments. 1935.

[9] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: evaluating models of vehicular environmental impact. In *SIGSPATIAL/GIS*, pages 269–278, 2012.

[10] M. Hua and J. Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *EDBT*, pages 347–358. ACM, 2010.

[11] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *ICDE*, pages 10–10. IEEE, 2006.

[12] H.-P. Kriegel, M. Renz, and M. Schubert. Route skyline queries: A multi-preference path planning approach. In *ICDE*, pages 261–272. IEEE, 2010.

[13] S. Kullback. *Information theory and statistics*. Courier Dover Publications, 1997.

[14] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[15] J. Letchner, J. Krumm, and E. Horvitz. Trip router with individualized preferences (trip): Incorporating personalization into route planning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1795. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[16] W. Luo, H. Tan, L. Chen, and L. M. Ni. Finding time period-based most frequent path in big trajectory data. In *SIGMOD*, pages 713–724. ACM, 2013.

[17] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *SIGSPATIAL*, pages 336–343. ACM, 2009.

[18] S. Rogers and P. Langley. Personalized driving route recommendations. In *Proceedings of the American Association of Artificial Intelligence Workshop on Recommender Systems*, pages 96–100, 1998.

[19] B. Yang, C. Guo, and C. S. Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *PVLDB*, 6(9):769–780, 2013.

[20] B. Yang, C. Guo, and C. S. Jensen. Travel cost inference from sparse, spatio-temporally correlated time series using markov models. *PVLDB*, 6(9):769–780, 2013.

[21] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang. Stochastic skyline route planning under time-varying uncertainty. In *ICDE*, pages 136–147, 2014.

[22] B. Yang, M. Kaul, and C. S. Jensen. Using incomplete information for complete weight annotation of road networks. *IEEE Trans. Knowl. Data Eng.*, 26(5):1267–1279, 2014.

[23] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*, pages 99–108. ACM, 2010.

[24] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Ubicomp*, pages 89–98, 2011.