2015 16th IEEE International Conference on Mobile Data Management

Vehicle Routing With User-Generated Trajectory Data

Vaida Čeikutė
Department of Mathematics and Informatics, Vilnius University
vaida.ceikute@mif.vu.lt

Christian S. Jensen
Department of Computer Science, Aalborg University
csj@cs.aau.dk

***Abstract*—Rapidly increasing volumes of GPS data collected from vehicles provide new and increasingly comprehensive insight into the routes that drivers prefer. While routing services generally compute shortest or fastest routes, recent studies suggest that local drivers often prefer routes that are neither shortest nor fastest, indicating that drivers value route properties that are diverse and hard to quantify or even identify. We propose a routing service that uses an existing routing service while exploiting the availability of historical route usage data from local drivers. Given a source and destination, the service recommends a corresponding route that is most preferred by local drivers. It uses a route preference function that takes into account the number of distinct drivers and the number of trips associated with a route, as well as temporal aspects of the trips. The paper provides empirical studies with real route usage data and an existing online routing service.**

I. Introduction

Widespread use of mobile devices, such as smartphones and navigation devices, enables the accumulation of large volumes of GPS data. This geographical user-generated content holds huge potential for use in different types of mobile services.

We provide and evaluate concepts and techniques that enable the use of GPS data collected from the drivers as input to a *local driver behavior* based routing service. We aim to provide users with routes that are commonly used by local drivers. Local drivers often have in-depth knowledge of the surrounding area and therefore can recommend better and more convenient routes.

Consider the four possible routes between source location S and destination location D shown in Figure 1. Each route

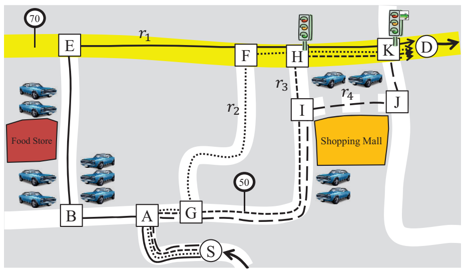


Figure 1. Motivating Example

has desired properties. Route r_1 (solid line) contains the longest part of the road with the highest speed, making it the fastest route. Route r_3 (dashed line) is shortest. Route r_2 (dotted line) avoids the parking lots near *Food Store* and *Shopping Mall* that may slow down the travel due to cars entering and leaving the parking lots, especially during *peak* hours. Route r_4 (long dashes line) can be attractive during *off-peak* hours when the stores are closed. It avoids one traffic light (intersection H) and passes intersection K with a "green" right turn signal. This example illustrates how some routes can be better than others at different times.

At the time when conventional routing services were created, no scalable means of collecting information such as that indicated in the example were available. This is changing: With the availability of GPS data volunteered by drivers, we are able to gain insight into the behaviors of drivers. This data can provide insight into the geographical aspects of drivers' choice of routes and can also reveal temporal behavior changes across a day, the days of the week, or even the seasons of the year.

Some existing studies report on the use of GPS data in routing services [5], [7], [11], [22]. Proposed routes are formed from parts of the road network that are covered by available GPS data by combining parts of different trajectories into one route. Most of the studies focus on inventing and evaluating algorithms that are efficient, and only few proposals evaluate their results in terms of quality. For example, few consider whether people actually use routes suggested by scoring functions. In real life, such recommended routes may not be attractive, and drivers may prefer different route between specific source/destination pairs than those formed from parts of trajectories.

Many factors can influence a user's route preference, and it is challenging to provide a user with the most convenient route because opinions on "what is convenient" may differ among users. Conventional routing services generally interpret convenience as either the fastest or the shortest route. This makes it easy to quantify the result. However, a recent study [19] shows that drivers do not necessarily follow the routes provided by routing services. Further analysis shows that preferred routes are not necessarily the shortest or fastest ones. As a result, we have no precise specification of the properties that characterize a preferred route.

In this paper, we consider routes formed from trajectories. Trajectories that follow the same paths are grouped into routes that are identified using the concept of Longest Common Subsequence (LCSS) [20]. With the help of a flexible scoring function, a preferred route is identified from the set of available routes. Our approach utilizes available trajectory data, and the scoring function scores a route using the number of available traversals of the route and the number of distinct drivers following the route, and it also considers temporal aspects. These properties of a route are readily available the from trajectory data. In practice, it is important to consider the number of distinct drivers because routes taken by multiple drivers are generally more attractive than routes traversed multiple times by one driver. Temporal aspects are also important because drivers may take different routes depending on the time of the day or the day of the week. If no data is available for a considered time interval, data from other time intervals should be used so that all available trajectory data is used to identify preferred routes.

When a new user issues a query for locations not covered by available GPS data, an available online routing service is simply used to provide the user with the route. This ensures that route can always be provided.

We present a careful, in-depth experimental study with high-quality, real-world data to obtain detailed insight into key aspects of our proposal. The study offers insight into route recommendation in different settings, and it also covers comparisons with competing routing techniques. The results show that the routes provided by the paper's routing service that uses GPS data from local drivers are better than those provided by competing routing services.

In summary, the main contributions of the paper are:

- We propose a routing service that utilizes GPS data from local drivers.
- We propose a flexible scoring function for route recommendation. Specifically, the numbers of traversals and distinct drivers, and temporal aspects of the routes are used.
- We report findings from an extensive study with real data collected during a two-year period from 285 drivers.
- We evaluate the quality of our proposed routes by simulating a feedback loop from the drivers.

The remainder of the paper is organized as follows. Section II provides the main definitions and presents the proposed system framework and algorithms. Section III reports on the results of the empirical study. Section IV covers related work, and Section V concludes the paper.

II. Routing with User-Generated Trajectories

A. Definitions

A GPS log is a collection of GPS points $pt_i = (p_i, t_i)$, where $p_i = (x_i, y_i)$ is a location in two-dimensional Euclidean space $\mathbb{R}^2$ and $t \in T$ is a timestamp. GPS locations are collected from a number of users $u \in U$ that are uniquely identified by an id. A location where a user stays for a certain duration of time is called a *stay point*. A sequence of GPS points reported by a user u that is between two *stay points* is considered as a separate trip, $trp = (u, plt) \in TRP$, where $plt = \langle pt_1, \ldots, pt_n\rangle$ is a polyline defined by a sequence of timestamped points $pt_i = (p_i, t_i)$, where $p_i \in P$ and $t_i \in T$. Here, $n \geq 2$ and $t_i < t_{i+1}$, $i = 0, 1, \ldots, n-1$.

The movements of users are constrained to a road network. A digital road network is formed from a logically linked set of polylines. The connectivity of the road network is represented by a graph $G = (I, S)$ that is formed from a set of intersections $I \subset P$ and a set of segments S. Each road segment is represented by a polyline pl that is formed by a sequence of points $s = \langle p_1, \ldots, p_n\rangle \in S$. A polyline approximates the centerline of a physical road. The part of a two-way road between a pair of intersections is represented by two polylines with reverse sequences of points.

We utilize GPS logs to identify the trips taken by a set of drivers throughout a monitoring period. Using map-matching, sequences of GPS points, $plt = \langle pt_1, \ldots, pt_n\rangle$, are transformed into sequences of road segments with timestamps capturing when the road segments were entered and exited, i.e., $sts = \langle st_1, \ldots, st_m\rangle \in STS$, where $st = (s, t_s, t_e), s \in S$, and $t_s, t_e \in T$.

After applying map-matching, trips are represented according to Definition 1.

Definition 1: A traversal is a 3-tuple $tr = (u, day, sts) \in TR$, where $u \in U$, $day \in [1, 7]$ identifies a day of the week, and $sts \in STS$.

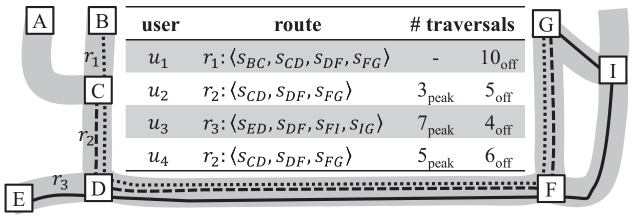
Each traversal occurs on a specific time of the day and day of the week. To capture the temporal aspect of route use, we group traversals according to the intersection of their starting times with a set of temporal intervals that we call *temporal patterns*. We divide one calendar day into three temporal patterns: peak-morning, peak-afternoon, and off-peak hours. A week is divided into weekdays and weekend days (including public holidays). Each weekday has all three temporal patterns, whereas weekend days are off-peak.

Definition 2: A temporal pattern $tp \in TP$ is defined as $tp = (TT, days)$. Here TT is a set of time intervals $tt = [t_s, t_e], t_s, t_e \in T$ that identify the start and end times of the pattern, and $days \subseteq [1, 7]$ represents the days of the week when the pattern is valid.

Traversals that follow the same sequence of road segments are joined into routes. Further, use two representations of routes. We represent a route as a sequence of road segments $pls = \langle s_1, \ldots, s_m\rangle \in PLS, s_i \in S$; and we represent a route as a sequence of geographic points that are formed from the geographical representations of road segments, $pl = \langle p_1, \ldots, p_n\rangle \in PL, p_i \in P$, i.e., polylines. For easy reference, both representations are used in the definition of a *route usage object*.

Definition 3: A route usage object $r \in R$ is defined as $r = (pls, pl, RTP)$, where $pls \in PLS$, $pl \in PL$, and $RTP = \{(tp, UA)_i\}$ is a set of temporal patterns together with a set of tuples $UA = \{(u, amnt)_j\}$ that provide the set of users and numbers of times each user has taken the route.

Example 1: Figure 2 gives an example of three routes along with a table providing traversal information. Assume that a user wants to travel from location D to location G during *peak* hours. In the example, all three routes contain D and G. Route r_1 is taken by user



user	route	# traversals	
u_1	r_1: $\langle s_{BC}, s_{CD}, s_{DF}, s_{FG}\rangle$	-	10_{off}
u_2	r_2: $\langle s_{CD}, s_{DF}, s_{FG}\rangle$	3_{peak}	5_{off}
u_3	r_3: $\langle s_{ED}, s_{DF}, s_{FI}, s_{IG}\rangle$	7_{peak}	4_{off}
u_4	r_2: $\langle s_{CD}, s_{DF}, s_{FG}\rangle$	5_{peak}	6_{off}

Figure 2. Route Score Calculation Example

u_1 10 times during off-peak hours. Route r_2 is taken by users u_2 and u_4 3 and 5 times during peak hours and 5 and 6 times during off-peak hours, respectively. These two routes share the same sequence of segments between the start and end locations. Thus, a new route usage object is formed: $r' = (\langle s_{DF}, s_{FG}\rangle, pl', RTP')$, $RTP' = \{(peak, \{(u_2, 3), (u_4, 5)\}), (off, \{(u_1, 10), (u_2, 5), (u_4, 6)\})\}$. Route r_3 is taken by user u_3 7 times during peak hours and 4 times during off-peak hours. Another route usage object is formed: $r'' = (\langle s_{DF}, s_{FI}, s_{IG}\rangle, pl'', RTP'')$, where $RTP'' = \{(peak, \{(u_3, 7)\}), (off, \{(u_3, 4)\})\}$.

Routes provided by an available online routing service are formed by a sequence of points from the geographical representations of the roads that belong to the routes, $rn = \langle p_1, \ldots, p_n \rangle \in RN \subset PL$.

B. Architecture

The architecture of the system is shown in Figure 3. In the online mode, a user issues a request to the server, providing source and destination locations along with a value k indicating the number of desired results. The server uses an available trajectory data set to identify the top-k preferred routes that are then returned to the user. If no available trajectories contain the source–destination (s–d) pair then a request is issued to an available online routing service, and the results from there are provided to the user.

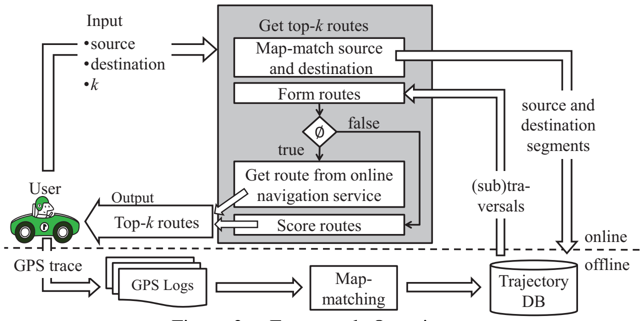


Figure 3. Framework Overview

While the user follows the route, the user's GPS trace is being logged and reported to the server. Then, when the trip is completed, in the offline mode, the trip is map-matched to the road network and stored in the trajectory data set for later use.

C. Scoring of Route Usage Objects

In our study, we assume that each traversal starts at the beginning and ends at the end of a road segment. The route usage objects that start and end on the same pair of segments are then possibly different routes between the same two locations. We introduce a scoring function that makes it possible to rank such routes.

We aim to present users with the routes that best reflect the preferences of local drivers. A recent study [19] shows that drivers favor routes that are not necessarily the shortest or fastest. We distinguish between two different properties that can influence the score of a route usage object, namely its popularity and its temporal fit. As discussed earlier, some routes may be more popular on specific times of the day or days of the week than at other times and days.

The number of users and the number of traversals that influence the popularity are totaled according to the temporal patterns they belong to. Given a temporal pattern tp_q $(\exists i(t_q \in tp_q.tt_i) \wedge day_q \in tp_q.days)$, where t_q and day_q are the query time and day, respectively, the number of users ($\text{users}(r)$) and the number of traversals ($\text{traversals}(r)$) are calculated as follows.

$$\text{users}(r) = \begin{cases} r.rtp_i.|UA|, & \text{if } tp_q = r.rtp_i.tp \\ \displaystyle\sum_{rtp \in r.RTP(r.rtp.tp \neq tp_q)} rtp.|UA|, & \text{otherwise} \end{cases}$$

$$\text{traversals}(r) = \begin{cases} \displaystyle\sum_{ua \in r.rtp_i.UA} ua.amnt, & \text{if } tp_q = rtp_i.tp \\ \displaystyle\sum_{rtp \in r.RTP(rtp.tp \neq tp_q)} \sum_{ua \in rtp.UA} ua.amnt, & \text{otherwise} \end{cases}$$

We quantify the popularity of a route based on its user count and based on how many times each user traversed the route. These counts are influenced strongly by how well the available GPS data covers the roads. To contend well with cases where, in the available GPS data, some users have traversed a route considerably more times than other users, we consider only those traversals that are less or equal to the *minimal* average number of user traversals per *available* route between the source and destination, i.e., $\text{avg}^{min}(r) = \frac{\text{traversals}(r)}{\text{users}(r)}$. Then, the preference function is given as follows:

$$\text{pref}(r) = \alpha \cdot \text{users}(r) + (1 - \alpha) \cdot \text{traversals}^{AVG}(r),$$

where $\alpha \in [0, 1]$. Depending on the value of α, the number of traversals or the number of users have a higher influence on the *preference* value. Function $\text{traversals}^{AVG}(r)$ gives the number of traversal that satisfy the additional case, i.e., $\forall ua \in r.rtp_i.UA(ua.amnt \leq \text{avg}^{min}(r))$.

Now we are able to define the scoring function for route usage objects.

Definition 4: Given a query temporal pattern tp_q, a start segment s_s, an end segment s_d, and a route $r \in R$ $(r.pls.s_1 = s_s \wedge r.pls_m = s_d)$, the score of the route r is:

$$\text{score}(r) = \beta \cdot \text{pref}^M(r) + (1 - \beta) \cdot \text{pref}^N(r),$$

where $\text{pref}^M(r)$ is the preference value of route r calculated using traversals with temporal pattern tp_q and $\text{pref}^N(r)$ is the preference value calculated using traversals with other temporal patterns.

Example 2: Consider again the example in Figure 2. The average value of traversals during different time patterns for routes r' and r'' are: $\text{avg}(r')_{peak} = \frac{3+5}{2} = 4$, $\text{avg}(r')_{off} = \frac{10+5+6}{3} = 7$, $\text{avg}(r'')_{peak} = 7$, $\text{avg}(r'')_{off} = 4$. Thus, the *minimal* average values that are used in the score calculation are $\text{avg}^{min}_{peak} = 4$, $\text{avg}^{min}_{off} = 4$. Then, with $\alpha = 0.5$, the preference value for matched and unmatched traversals of route r' are $\text{pref}^M(r') = 0.5 \cdot 2 + 0.5 \cdot (3 + 4) = 4.5$ and $\text{pref}^N(r') = 0.5 \cdot 3 + 0.5 \cdot (4 + 4 + 4) = 7.5$. Let us prioritize the traversals that match the temporal pattern in the scoring function by setting $\beta = 0.75$. Then the score of route r' is $\text{score}(r') = 0.75 \cdot 4.5 + 0.25 \cdot 7.5 = 5.25$. Similarly, the score of route r'' is $\text{score}(r'') = 0.75 \cdot 2.5 + 0.25 \cdot 2.5 = 2.5$. Because $\text{score}(r') > \text{score}(r'')$, route r' is the most preferred by the users in the available dataset, and it is the first route suggested to a user.

D. Route Identification

A set of traversals are joined into *route usage objects*. To identify which traversals represent the same route, we use the notion of Longest Common Subsequence (LCSS) [20]. Consider two trajectories represented by sequences of road segments: $pls = \langle s_1, \ldots, s_n \rangle$ and $pls' = \langle s'_1, \ldots, s'_m \rangle$. The longest common subsequence $\mathrm{LCSS}(pls, pls')$ is defined as follows:

$$\mathrm{LCSS}(pls, pls') =$$

$$\begin{cases} 0, \text{ if } n = 0 \vee m = 0 \\ \mathrm{LCSS}(pls_{i-1}, pls'_{j-1}) + 1, \text{ if } s_i = s'_j \\ \max\{\mathrm{LCSS}(pls_i, pls'_{j-1}), \mathrm{LCSS}(pls_{i-1}, pls'_j)\}, \text{ otherwise} \end{cases}$$

We then define the similarity S_1 between two trajectories pls and pls':

$$\mathrm{S}_1(pls, pls') = \frac{\mathrm{length}(\mathrm{LCSS}(pls, pls'))}{\mathrm{length}(pls)} \cdot 100\%$$

Here, $\mathrm{length}(\cdot)$ returns the length of the argument polyline. The match between a pair of traversals thus depends on the length of the traversal. If a trip is long, the similar part has to be larger.

Whether two trajectories pls and pls' represent the same route is decided by the following predicate that uses Algorithm 1:

$$\mathrm{C}_1(pls, pls') = \begin{cases} true, \text{ if } \mathrm{S}_1(pls, pls', \epsilon) \geq \mathrm{getMatch}(pls, pls') \\ false, \text{ otherwise} \end{cases}$$

Algorithm 1 $\mathrm{getMatch}(pls, pls')$

```
IN: $pls, pls' \in PLS$
OUT: $sim \in \mathbb{R}$
1: $len_{avg} \leftarrow (\mathrm{length}(pls) + \mathrm{length}(pls'))/2$;
2: $sim \leftarrow 100\% - \max\{sim_{min}, (100 - len_{avg}) \cdot sim_{max}/100\}$;
3: return $sim$;
```

Parameters $sim_{min} \in \mathbb{N}$ and $sim_{max} \in \mathbb{N}$ are used to calculate the similarity value. The possible percentage difference between two polylines is in the range $[sim_{min}, sim_{max}]$, the average length of polylines pls and pls' is also taken into account (line 2).

To determine whether the route that the driver takes and the route recommended by the routing service are the same, LCSS is evaluated using the trajectories represented by a sequence of points. Consider two such polylines: $pl = \langle p_1, \ldots, p_n \rangle$ and $pl' = \langle p'_1, \ldots, p'_m \rangle$. Given an integer distance threshold ϵ, the longest common subsequence $\mathrm{LCSS}_\epsilon(pl, pl')$ is defined as follows:

$$\mathrm{LCSS}_\epsilon(pl, pl') =$$

$$\begin{cases} 0, \text{ if } n = 0 \vee m = 0 \\ \mathrm{LCSS}_\epsilon(pl_{i-1}, pl'_{j-1}) + 1, \text{ if } \mathrm{distance}(pl, p'_j) \leq \epsilon \\ \max\{\mathrm{LCSS}_\epsilon(pl_i, pl'_{j-1}), \mathrm{LCSS}_\epsilon(pl_{i-1}, pl'_j)\}, \text{ otherwise} \end{cases}$$

Recall that the geographical representation of a route is obtained by concatenating the polylines from the sequence of segments. The polyline that represents a route returned by an existing routing service contains key points of the road polyline using possibly another digital map. The key points in different maps may not match. Therefore, LCSS is evaluated using a $\mathrm{distance}(pl, p)$ function that gives the Euclidean distance from point p to polyline pl. Other similarity constraints have to be satisfied, as discussed above, for trajectories represented as sequences of road segments.

The general procedure for grouping GPS traversals into routes during the route identification step is presented in Algorithm 2. The algorithm takes a set of traversals and source and destination segments as input and returns a set of *route usage objects*.

Algorithm 2 $\mathrm{getRoutes}(TR_{in}, s_s, s_d)$

```
IN: $TR_{in} \subseteq TR, s_s, s_d \in S$
OUT: $R_{out} \subseteq R$
1: $R_{out} \leftarrow \emptyset$;
2: for $tr \in TR_{in}$ do
3:     $pls' \leftarrow \mathrm{getSegmentSeq}(tr.sts)$;
4:     if $\exists r \in R_{out}(\mathrm{C}_1(r.pls, pls') = true)$ then
5:         update temporal patterns set $r.RTP$;
6:     else
7:         $pl \leftarrow \perp$;
8:         while $s \leftarrow \mathrm{getFirst}(pls') \neq \perp$ do
9:             $pl' \leftarrow s.pl$;
10:            if $pl \neq \perp \wedge \mathrm{lastPoint}(pl) \neq \mathrm{firstPoint}(pl')$ then
11:                $pl \leftarrow \mathrm{concat}(pl, pl')$;
12:            else
13:                $pl \leftarrow \mathrm{concat}(pl, \mathrm{getTail}(pl'))$;
14:            end if
15:        end while
16:        $tp \leftarrow \mathrm{getTempPatt}(tr.sts_1.t_s, tr.day)$;
17:        $r \leftarrow (pls, pl, \{(tp, \{(tr.u, 1)\})\})$;
18:        $R_{out} \leftarrow R_{out} \cup r$;
19:    end if
20: end for
21: return $R$;
```

First, the set of route usage objects is initialized to the empty set (line 1). Then, each traversal $tr \in TR$ is processed (lines 2–20). A full sequence of segments (without start and end times) is stored in pls' (line 3). Then, if a route exists with the same sequence of segments, i.e., the route satisfies the similarity requirements, information about the route is updated (lines 4–5). Depending on the length of the route, the similarity that has to be satisfied between two polylines varies from 1 to 10 percent.

If traversal tr is formed from a new sequence of segments, a new route object is created and added to the result set (lines 6–19). In this case, first the geographical representation of the route is formed by concatenating the polylines from the sequence of segments pls' of traversal tr' (lines 7–15). There can be cases where segments are very short and no GPS points are reported while they are being traversed. In map matching, depending on whether the first point of the next segment in sts' matches the last point of the previous segment, it is either included in or discarded from the resulting polyline pl (lines 10–14). Then the temporal pattern tp that traversal tr belongs to is identified, and the new route usage object r is added to the output set (lines 16–18).

When each traversal $tr \in TR$ is processed, a final set of route usage objects R_{out} is returned (line 21).

E. Route Search Algorithm

Algorithm 3 identifies the top-k routes with the highest scores. As input, it takes a start location p_s, a destination location p_d, a time of day t_q, a day of the week day_q, an existing traversal data set TR_{in}, and the value k that gives the number of desired results.

Algorithm 3 getTopKRoutes($p_s, p_d, t_q, day_q, TR_{in}, k$)

```
IN: $p_s, p_d \in P, t_q \in T, day_q \in \mathbb{N}, TR_{in}, k \in \mathbb{N}$
OUT: $R_k \subseteq R$
1: $TR' \leftarrow \emptyset$;
2: $s_s \leftarrow$ getSegment($p_s$); $s_d \leftarrow$ getSegment($p_d$);
3: for $tr \in TR$ such that $\exists i, j\ (i \leq j \wedge tr.sts_i = s_s \wedge tr.sts_j = s_d)$ do
4:     $sts' \leftarrow \langle tr.sts_i, \ldots, tr.sts_j \rangle$;
5:     $tr' \leftarrow (tr.u, tr.day, sts')$;
6:     $TR' \leftarrow TR' \cup tr'$;
7: end for
8: $R' \leftarrow$ getRoutes($TR'$);
9: $R_k \leftarrow$ score routes in $R'$;
10: return $R_k$;
```

First, the set of intersecting traversals is initialized as the empty set (line 1). Then the start p_s and destination p_d locations are map-matched to road segments (line 2), upon which the set of traversals that contain the source and destination segments s_s and s_d are stored in set TR' (lines 3–7). Every such traversal is trimmed to be between segments s_s and s_d (line 4). A new traversal tr' is created and stored in the set TR' (lines 5–6). From the set of new traversals, a set of routes R' is identified using Algorithm 2 (line 8). Each of these routes is scored using the scoring function, and the k routes with the highest scores are returned (lines 9–10).

III. Experimental Study

This section presents an evaluation of the paper's proposal. First, we describe the data set used, then we explain how the experiments were conducted, and finally we present the experimental findings.

A. Data Description

In the experiments, we use a high-quality real GPS data set collected at 1Hz from 285 distinct drivers who took part in the "Pay as You Speed" project[1]. The drivers were monitored during a two-year period. As all the drivers were residents of North Jutland, Denmark, the GPS data offers relatively dense coverage of this part of Denmark.

This data set was preprocessed with a modified M-GEMMA map-matching algorithm [2], [13]. Each trip in the resulting data set contains at least 10 GPS recordings and one road segment. If there is only one single segment between two identified trips and less than 120 seconds between their timestamps then these trips are joined into one. We discarded trips that were less than 2 kilometers long. The final data set contains 182,714,637 GPS records and 247,713 trips that cover a total of 2,749,303 kilometers. Figure 4 gives a brief overview of the available traversals according to their lengths and temporal periods. As is shown, more than 90% of the traversals are between 2 and 20 kilometers long. The average length of a trajectory is 10 kilometers. Also, 92% of the traversals is taken during off-peak hours.

We used OpenStreetMap[2] for Denmark. The map contains 659,015 road segments. Our data set covers 152,773 distinct

[1]www.sparpaafarten.dk

[2]www.openstreetmap.org

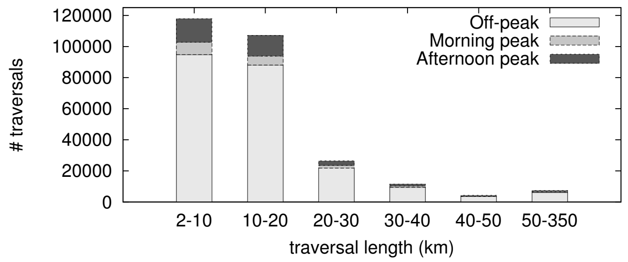


Figure 4. Overview of Traversals

road segments, i.e., 23% of the map. A brief overview of the popularity of the segments in the road map is given in Table I. It is shown that 50% of the covered segments were

Table I
The Popularity of Segments

# traversals	# segments
1–10	68,063
10–100	62,369
100–1K	19,160
1K–3K	2,705
3K–10K	476

traversed from 10 to 10K times. Another half of segments were traversed between 1 and 10 times.

B. Experimental Settings

To evaluate the quality of the routes returned by our proposal, we conduct a series of experiments with a simulated feedback loop. For each test case, we select a subset of traversals from the available trajectory data set that fits the purpose of the particular test case. The remaining data is used in the route identification process.

Data Sets for Testing

For the experiments, we use two subsets of our full trajectory data set. The first data set (*Length*) consists of randomly selected trips with different lengths. These traversals originate and end at source–destination (s–d) pairs that are used at least twice in the full data set. This condition is chosen in order to ensure that there is always a possible route to recommend in response to a query. Each source–destination pair appears once in the *Length* data set. This allows us to explore diverse trips in the experimental study. To select test trips for the *Length* data set, we first divided the full trajectory data set into three sets consisting of: short (2–10 km), medium (10–20 km), and long (20–350 km) trajectories. We then selected equal amounts of test trips from these three sets at random. Figure 5 gives an overview of the amounts of test trips per length interval during specific temporal periods.

For the second data set (*Driver*), we chose the 5 drivers with the most trips. An overview of the data according to length and temporal period is provided in Figure 6(a) and Figure 6(b), respectively. This subset of the data contains mostly short traversals, between 2 and 10 kilometers, and most of these traversals were taken during off-peak hours. Recall that the same tendency is also noticed in the full trajectory data set (see Figure 4).

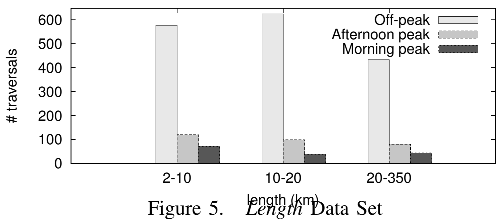
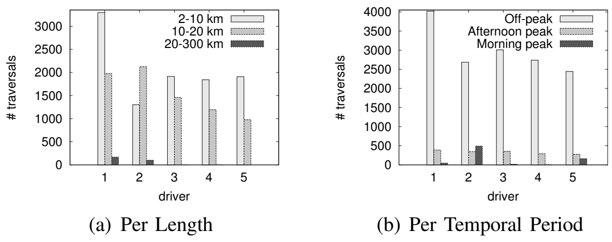


Figure 5. *Length* Data Set



(a) Per Length (b) Per Temporal Period

Figure 6. *Driver* Data Set

The numbers of traversals per driver are given in Table II along with the numbers of distinct source–destination pairs. Figure 7 provides an overview of the amount of source–destination pairs in the *Driver* data set that were used more than nine times. It follows that this trajectory data set contain

Table II
TRAVERSALS OF *Driver* DATA SET

Driver	# traversals	# distinct s–d pairs
1	4458	1552
2	3529	1705
3	3377	1582
4	3039	1586
5	2885	1406

trips that originate and end at specific source–destination pairs that are used multiple times (not only once). For example, Driver 1 and Driver 2 have s–d pairs that were used some 200 times during the monitoring period. For Drivers 3, 4, and 5, the most frequently visited s–d pair occurs in some 50 trips.

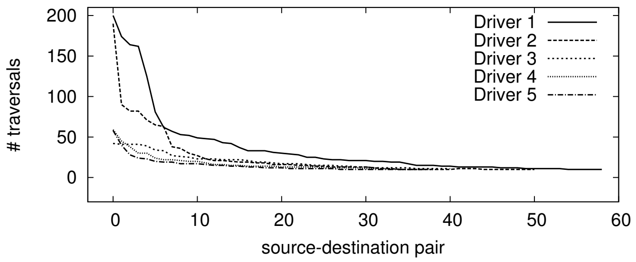


Figure 7. # s–d pairs vs. #traversals$\geq$10

Test Procedure

The procedure for evaluating the quality of a preferred route is presented in Algorithm 4. The procedure simulates a feedback loop from drivers.

First, three counters used for statistics of the results are initialized to zero (line 1). Counter $match_{cnt}$ stores the number of *matched* trips, i.e., trips that satisfy the defined similarity measures (the test trip follows the same route as the preferred route). Counter not_{cnt} stores the number of *unmatched* test trips that follow a different route than the preferred route. The last counter, $empty_{cnt}$, stores the number of *empty* trips, i.e., the number of times where our framework provided an empty result, meaning that the available trajectory data set does not contain the provided source and destination locations. Next, a subset of the full trajectory data set is selected for testing (line 2). Each

Algorithm 4 doTest()

```
1: $match_{cnt} \leftarrow 0$; $not_{cnt} \leftarrow 0$; $empty_{cnt} \leftarrow 0$;
2: $TR_t \leftarrow (TR \backslash TR_{Length}) \vee (TR \backslash TR_{Driver})$;
3: for $tr_t \in TR_t$ do
4:     $t_q \leftarrow sts_t.st_1.t_s$; $sts_t \leftarrow tr_t.sts$;
5:     $p_s \leftarrow sts_t.st_1.s.p_1$; $p_d \leftarrow sts_t.st_m.s.p_n$;
6:     $R_k \leftarrow$ getTopKRoutes($p_s, p_d, t_q, tr_t.day, TR_t, 1$);
7:     if $R_k = \emptyset$ then
8:         $empty_{cnt} \leftarrow empty_{cnt} + 1$;
9:     else
10:        $pls_r \leftarrow r_1.sts(r_1 \in R_k)$;
11:        $pls_{tr} \leftarrow$ getSegmentSeq($tr_t.sts$);
12:        if $C_1(pls_r, pls_{tr}) = true$ then
13:            $match_{cnt} \leftarrow match_{cnt} + 1$;
14:        else
15:            $not_{cnt} \leftarrow not_{cnt} + 1$;
16:        end if
17:    end if
18:    $TR_t \leftarrow TR_t \cup tr_t$;
19: end for
```

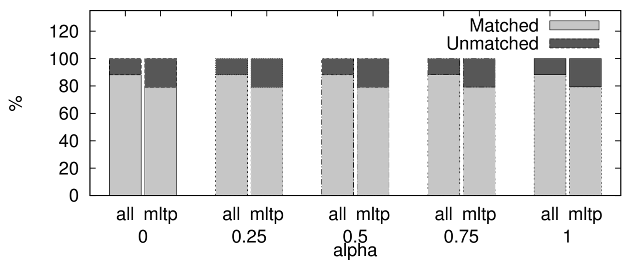
traversal in the test subset TR_t is processed (lines 3–19). For each test traversal tr_t, the start time t_q, start point p_s, and the destination point p_d are extracted (lines 4–5). In line 6, the top–1 preferred route is found using getTopKRoutes. If no preferred route is found, the value of counter $empty_{cnt}$ is incremented. Otherwise, the segment sequences representing the preferred route r_1 and the test traversal tr_t are compared, and, depending on the result, the corresponding counter values are incremented.

C. Experimental Results

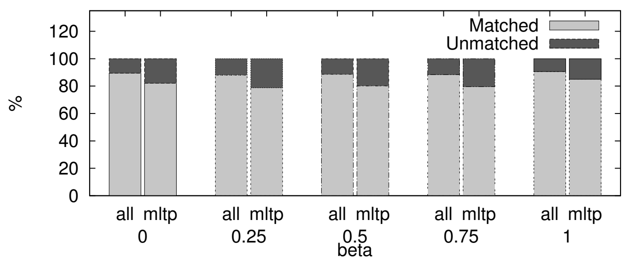
Influence of α and β

The first experiment is designed to identify the best values of α and β for our data set.

A higher α value decreases the influence of the number of traversals on the route score. For the α experiment, we set $\beta = 0.95$, so that matched traversals influence the score more and so that there is always a recommended route, even if there does not exist traversals that fit the temporal pattern of the test trip. This way, we make sure that a preferred route is found and that the *empty* case does not occur. The results are provided in Figure 8(a) (bars under label "all"). It can be seen that the α value does not influence the resulting amount of matched routes. There are several reasons for this: (a) in most cases, only one route is recommended to the user, (b) only few distinct drivers are taking the route, and (c) the number of traversals is low.



(a) Varying α ($\beta = 0.95$)



(b) Varying β ($\alpha = 0.5$)

Figure 8. *Length* Data Set Analysis

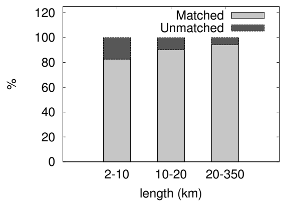
For the β experiment, we set $\alpha = 0.5$. The results show the same tendency as in the α experiment (see Figure 8(b), bars under label "all"). A varying β value has low influence on the number of matched trips.

If there exists only one possible route between a source–destination pair, the α and β values do not matter for the final count of *matched* and *unmatched* test trips. Thus, the second bar in Figure 8(a) and 8(b), labeled "mltp", shows the count of test trips for which we identified several alternative routes between the source–destination pairs. In the *Length* data set, there are 871 such trips out of 2,086 trips. For varying α values, the same tendency is observed as for the "all" trips count. Considering β, the highest count of matched trips is obtained with $\beta = 1$, i.e., the preferred route is formed only from traversals that were taken during the same time period. This means that the best preferred routes are selected when only the traversals that have the same temporal pattern as the test trips are considered.

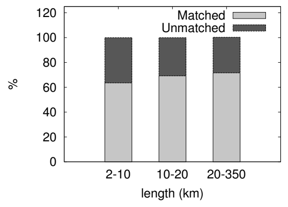
In the subsequent experiments, we use $\alpha = 0.5$ and $\beta = 0.5$. The value $\beta = 1$ was not chosen because we prefer, if possible, to always present the user with some route rather than returning no result.

Influence of Trip Length

We continue with an experiment that aims to help us understand how good the provided preferred routes are for different trip length intervals. Results are given in Figure 9(a). A higher percentage of matches are seen for longer trips (∼95%). For short trips of length 2–10 kilometers, only 80% of the trips are matched. This mirrors the fact that, in our data set, drivers follow more diverse routes for short trips than for long trips.



(a) Preferred Route

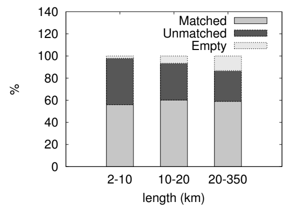


(b) Google Route

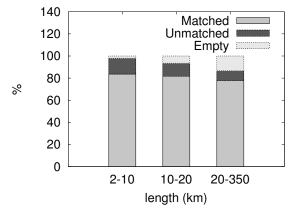
Figure 9. Match Per Trip Length (*Length*)

For the same set of test trips, we also check how they match the routes provided by the Google Directions API (see Figure 9(b)). This provides an understanding of how many trips can match when no preferred route is available in the trajectory data set. For each length interval, only 60–70% of the trips are matched. A higher percentage is observed for longer trips.

Further, we use the same test data set and consider the trips using the Time Period-Based Most Frequent Path (TPMFP) [11] approach (see Figure 10). This is the most recent study that exploits user-generated GPS data to score routes. In this approach, a so-called footmark graph is created from trips that fit the provided time period and have the given destination node. For cases where no traversals in our data set fall into the temporal period and have the required source and destination locations, it is not possible to provide the user with a route. Such a result is marked as "empty" in the graph.



(a) Exact Match



(b) Match With Similarity Measures

Figure 10. TPMFP Route (*Length*)

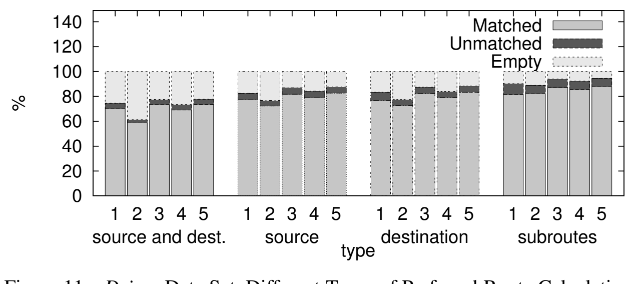
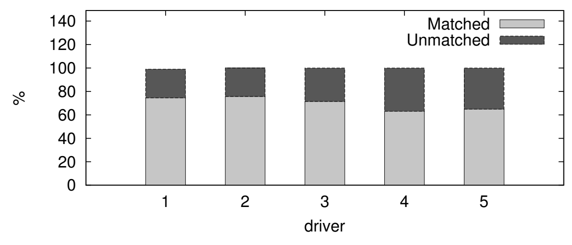
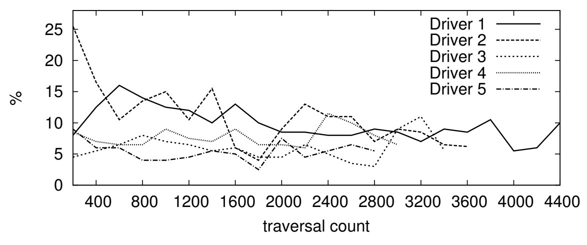


Figure 11. *Driver* Data Set, Different Types of Preferred Route Calculation

While comparing TPMFP with the route the test trip represents, we distinguish two cases: exact match (Figure 10(a)), and match with similarity measures (Figure 10(b)). The first case, exact match, means that the sequence of segments is the same. Here, for each length range, a match is identified for about 55–60% of the trips.



(a) Comparison With Google Routes



(b) Empty Routes

Figure 12. *Driver* Data Set Analysis

The highest percentage of trips (43%) that do not match is observed for trips with lengths in the range 2–10 kilometers. The results also show that due to the lack of GPS data, the highest percentage of empty results are for trips in the length range 20–350 kilometers. In the second case, we compare routes using the defined similarity measures, i.e., the sequences of segments are almost the same. Here, for each length range, 80% of the trips matches. Again, the highest percentage of unmatched trips occurs for the 2–10 kilometers length range. When comparing cases (a) and (b), we found that the most common error that TPMFP gives for recommended routes is a short detour along the route. These detours appear because the resulting route is formed from parts of several routes that have a higher traversal rate.

From this experiment, it can be concluded that routes recommended to users are taken more often by local drivers when they are formed from previous traversals of drivers that live in the same region than when they are provided by the Google Directions API. When identifying a preferred route, we make use of all trips, i.e., not only those that fit the temporal pattern. Therefore, for the *Length* data set, we do not get "empty" results as in the TPMFP approach. Considering the same similarity measures, we get similar result for trips in the 2–10 kilometers length range, but for the other two ranges, we get better percentages of matched trips.

The quality of a route is judged personally. It is difficult to guess why some particular route is more appealing in a particular region. But, assuming that local drivers have knowledge of the local region, local drivers' routes are most likely also attractive to tourists and visitors.

Driver Based Evaluation

This study does not aim to make general conclusions about all the drivers, but rather aims to offer insight into the extent to which our system is able to provide routes to these specific 5 drivers as new users, i.e., we consider the case where data collected from other drivers is used in the route identification process. Recall that for the test cases, we identify a preferred route using a subset of available trajectory data (see Algorithm 4). For this set of experiments, we used $\alpha = \beta = 0.5$.

As discussed in Section II-C, a preferred route is formed using available *full* routes and *sub*-routes. We use the term *sub-route* for a route that contains the source and destination, but may start or end at different locations.

First, we study how the use of sub-routes influences the preferred routes returned by the system. Figure 11 shows the results of matched, unmatched, and empty routes for different drivers using different data sets for preferred route identification.

It is expected that the first set of bars in Figure 11, labeled "source and dest.", have the most empty routes. Here, we require that the considered routes start and end at the source and destination locations given in the query. Therefore, the data considered is considerably smaller than when using sub-routes. With the restrictions that a route should only start (see the "source" bars) or end (see the "destination" bars), we see an improvement in the preferred route availability. The fewest empty routes occur when a preferred route is identified using all available sub-routes. It is also observed that the amount of *unmatched* routes increases when sub-routes are considered. We feel that the use of sub-routes represents a good trade-off considering the clear decrease in empty routes.

While comparing the routes that drivers are taking against the route provided by the Google Directions API (see Figure 12(a)), we see almost the same results as for the *Length* data set, i.e., 60–70% of the trips taken by drivers match the routes provided by Google.

Table III
Driver SET, ANALYSIS OF UNMATCHED TRIPS

Driver	**# traversals**	**input =** $true$	**input =** $false$
1	388	69%	31%
2	241	46%	54%
3	225	74%	26%
4	206	63%	37%
5	197	70%	30%

A preferred route is identified using past trips taken by the users of the system. Thus, we want to check how often a preferred route given to a driver is formed using traversals taken by the driver him- or herself. Table III gives details on the unmatched trips for the *Driver* data set. The first column identifies a test driver. The second column gives the number of unmatched cases for each driver. The third column provides the percentage of unmatched cases when the preferred route was formed using trips previously taken by the test driver considered (input=$true$). The fourth column provides the percentage of the opposite cases (the test driver did not previously take a preferred route). The table shows that in more than 50% of the cases (except for Driver 2), the preferred route was formed using the driver's own past trips. It thus follows that drivers change routes between specific source–destination pairs, i.e., a specific driver does not always take the same route.

Further we study how fast a preferred route is available to a new user in the system. For this experiment, we calculate the percentage of empty routes after every 200 trips for a specific driver. Results are given in Figure 12(b).

Driver 1 has a low percentage of empty routes for the first 200 traversals, but it increases substantially for the next 400 trips. Then a decrease is observed. Driver 2 first has ∼25% empty routes. The percentage subsequently varies in the range 10–15%. Between 1,800 and 2,200, the percentage starts to increase again. Driver 3 has peaks at 800 and 3,200, in-between which the percentage is steadily decreasing. For Driver 4, the empty route count varies in the range 6–9% until a peak is observed at 2,400; later, it again decreases. The steadiest graph is for Driver 5. One of the reasons why there can be an increase of empty preferred routes during the monitoring period is that drivers are going to new destinations that are not covered with available GPS data.

Figure 13 gives the results on matched and unmatched trips per driver when the full framework is used. Here, when a preferred route is not available using GPS data, a route is retrieved using the Google Directions API. A match is identified for more than 85% of the trips for each driver.

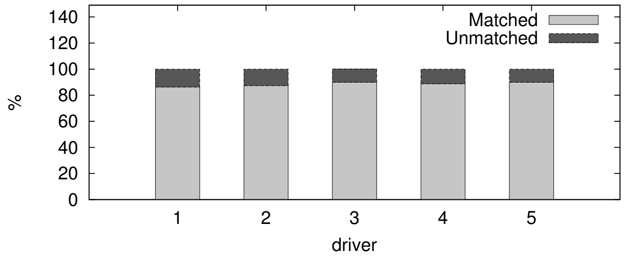


Figure 13. Results Using Proposed Framework

IV. Related Work

Trajectories in Routing Services

There have been several recent attempts at increasing routing quality by employing knowledge of previous trips.

Yuan et al. [22] propose a service that uses taxi trajectory data for route planning. This approach first constructs a time-dependent landmark graph, where a node (i.e., a landmark) is a road segment traversed frequently by taxis and an edge is a trajectory. Given start and destination locations, the two closest landmarks are identified, and a rough route (a sequence of landmarks) is found. Then a refined route (a concatenation of trajectories) connecting these landmarks is identified. Users are presented with the fastest route available. This study also includes a comparison with routes from Google Maps. It is found that when using roads that are used often by taxis, the travel time can be reduced by up to 26%. This study focuses on travel time; the actual routes taken are not studied. In contrast, our framework does not target fastest routes. Rather, we identify routes that are the most preferred by local drivers, i.e., have the most different drivers and have high numbers of traversals. To identify a preferred route, we also take into consideration when the query was issued, so that the temporal pattern of the suggested route also fits.

Chen et al. [7] present an algorithm that finds the most popular routes from one location to another using historical trajectory data. The authors suggest that this service is beneficial for tourists who are unfamiliar with the surroundings. The routes provided to the users are based on the travel behaviors found in the data set used. A transfer network is formed that contains *transfer nodes* that are significant locations for the users. These can be either starts or ends of trips, or they can be intersections of trajectories. A recommended route is formed from a sequence of transfer nodes that maximize the product of transfer probabilities. The authors use Absorbing Markov Chains to measure the popularity of a route. Our routes are formed from trips that contain paths between the source and destination, i.e., they are not formed from parts of multiple available routes, which we find can result in unattractive routes.

A framework presented by Chang et al. [5] focuses on the discovery of personalized routes. A road network is constructed from segments that are identified from a historical trajectory data set. Routes are synthesized by using the most frequently traveled road segments in the trajectory data set, and the top-k personalized routes are returned. If no trajectory connects two frequently used road segments, the shortest path between such segments is identified. In our study, we present users with top-k preferred routes that are generally not taken by the user. We also consider other drivers taking the route.

A recent study [11] provides a new scoring method to identify time-period based frequent paths. A route is formed from possibly multiple different routes. Trajectories used for scoring have to satisfy several constraints: they must contain the destination node and must also start and end during a provided time period. In contrast, we take parts of full trajectories that contain both the source and destination given by the user. This way, we make sure that a recommended route is one that has actually been used by drivers. We also do not only consider the number of available trajectories in the dataset when scoring a route, but consider also how many different drivers have taken the route.

Our experimental study includes comparisons with the most popular routes (MPR) [7], the shortest route, and the least number of road segments approach. The comparison is made using 1 month of data from multiple source locations to one destination location and encompasses 2,808 queries. The results show that 80% of the queries provide different results when using the different approaches. In our study, we compare our results (preferred routes) using a simulated feedback loop from the users, i.e., we compare with the paths the test user actually took. We chose this comparison to evaluate how good preferred routes are with respect to actual driver behavior. Comparison against other similar methods does not provide such input.

Trajectory Similarity

In our study, we use two representations of trajectories: sequences of roads segments and sequences of geographical points that form centerlines of road segments.

Existing approaches to trajectory comparison fall into three categories.

The first category is sequence based. Comparison based on Euclidean distance is sensitive to noise and cannot be applied to sequences with different lengths [4], [9]. The use of Dynamic Time Warping (DTW) distance [3], [8], [16] enables comparison between sequences of different lengths, but the mapping between sequences has to be continuous and monotonic. Since GPS trajectories can have gaps due to device malfunction, DTW approaches may not give good similarity results. Longest Common Subsequence (LCSS) [20], [21] is a more flexible similarity comparison technique because it allows to leave some points in the sequences unmatched. Edit Distance on Real sequences (EDR) [6] is a similar approach that takes into consideration gaps by assigning penalties. It is based on edit distance on strings and is suitable for comparison of trajectories with substantial noise.

The second category is shape-based similarity measures [1], [10], [12]. These methods ignore the ordering of the points in a trajectory. Alt et al. [1] use Fréchet distance to compare trajectories. Pelekis et al. [12] study geometric issues of trajectories and propose a set of trajectory distance operators based on space, time, speed, and direction.

The third category considers trajectory similarity under road network constraints [14], [15], [17], [18]. Here, raw trajectory data must be transformed into a form that ensures given motion restrictions. A spatial network can be represented as a directed graph. Therefore, a trajectory can be represented as a sequence of timestamped graph nodes [18] or a sequence of edges [14], [15], [17]. This typically calls for map-matching, which makes the quality of the results sensitive to the map-matching performance. Two network-represented trajectories match if the graph nodes or edges in the trajectories are in close proximity. Thus, trajectories do not have to follow the same exact routes to match.

We base our trajectory comparison on LCSS because it allows for gaps in the GPS data. We were satisfied with the quality of the results after extensive visual inspection.

V. Conclusions and Future Work

We present a routing service that utilizes trajectory data collected from local drivers for identifying preferred routes between source and destination locations. We present a set of algorithms that detail how to utilize GPS data for the route identification process.

The identified routes between a source–destination pair are scored, and the top-k preferred routes are returned to the user. We propose a flexible scoring function that takes into consideration the number of traversals of a route, the number of distinct drivers taking the route, and the time periods when the traversals occurred. Each of these characteristics influences the score of a route, and the degree of influence can be adjusted according to the needs of the routing service.

We present an in-depth experimental evaluation done using two different test data sets: a set of trips selected randomly from different length ranges, and a set of trips taken by several pre-selected drivers. Among other studies, we report on how often the provided preferred route matches the route that the driver actually took, and we also compare with the routes recommended by the Google Directions API and the TPMFP [11] approach. We find that 60–70% of the routes recommended by the Google Directions API match the users' actual choice of routes, while our approach reaches about 90%. The TPMFP approach achieves some 80%; and since this approach uses only the trips that fit the query's temporal pattern, no route is recommended for some 10% of the directions queries. Our approach avoids this drawback because it utilizes all the available data. We believe that this demonstrates that the proposal is effective and represents an improvement over the state-of-the-art.

The evaluation of the proposed framework shows that the use of user-generated GPS data can indeed improve the quality of a routing service. Thus, it is relevant to consider additional aspects of the framework. One such aspect is the efficiency of route identification process. Specifically, the LCSS technique. Another aspect is the inclusion of personalized routes into the system. It is possible that a specific driver prefers to take different routes during different times of the day. Yet another aspect is the support for routes that are constructed from sub-routes. This is of interest because it can yield many more routes for recommendation. However, there are cases where a route resulting from the concatenation of several routes is not unattractive. For example, the connectivity at road intersections has to be considered carefully. A good first step in this direction is to study different techniques experimentally.

Acknowledgments

We thank the reviewers for their helpful comments. The research was supported in part by a grant from the Obel Family Foundation.

References

[1] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the Fréchet distance. In *STACS*, pages 63–74, 2001.
[2] O. Andersen and K. Torp. An open-source ITS platform. Technical report, Aalborg University, 2012.
[3] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.
[4] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.
[5] K.-P. Chang, L.-Y. Wei, M.-Y. Yeh, and W.-C. Peng. Discovering personalized routes from trajectories. In *LBSN*, pages 33–40, 2011.
[6] L. Chen, M. Tamer Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.
[7] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *ICDE*, pages 900–911, 2011.
[8] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.
[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.
[10] B. Lin and J. Su. One way distance: For shape based similarity search of moving object trajectories. *GeoInformatica*, 12(2):117–142, 2008.
[11] 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, 2013.
[12] N. Pelekis, I. Kopanakis, I. Ntoutsi, G. Marketos, and Y. Theodoridis. Mining trajectory databases via a suite of distance operators. In *ICDE Workshops*, pages 575–584, 2007.
[13] F. Pereira, H. Costa, and N. Pereira. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review*, 1(3):107–124, 2009.
[14] G.-P. Roh and S.-W. Hwang. Tpm: supporting pattern matching queries for road-network trajectory data. In *EDBT*, pages 554–557, 2011.
[15] G.-P. Roh, J.-W. Roh, S.-W. Hwang, and B.-K. Yi. Supporting pattern-matching queries over trajectories on road networks. *TKDE*, 23(11):1753–1758, 2010.
[16] Y. Sakurai, M.i Yoshikawa, and C. Faloutsos. FTW: fast similarity search under the time warping distance. In *PODS*, pages 326–337, 2005.
[17] E. Tiakas, A. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *Journal of Systems and Software*, 82(5):772–788, 2009.
[18] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Trajectory similarity search in spatial networks. In *IDEAS*, pages 185–192, 2006.
[19] V. Čeikutė and C.S. Jensen. Routing service quality–local driver behavior versus routing services. In *MDM*, pages 97–106, 2013.
[20] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–685, 2002.
[21] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou. An effectiveness study on trajectory similarity measures. In *ADC*, pages 13–22, 2013.
[22] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *GIS*, pages 99–108, 2010.