2013 IEEE 14th International Conference on Mobile Data Management

Routing Service Quality—Local Driver Behavior Versus Routing Services

Vaida Čeikutė Christian S. Jensen
Department of Computer Science, Aarhus University, Denmark
{ceikute, csj}@cs.au.dk

***Abstract*—Mobile location-based services is a very successful class of services that are being used frequently by users with GPS-enabled mobile devices such as smartphones. This paper presents a study of how to exploit GPS trajectory data, which is available in increasing volumes, for the assessment of the quality of one kind of location-based service, namely routing services. Specifically, the paper presents a framework that enables the comparison of the routes provided by routing services with the actual driving behaviors of local drivers. Comparisons include route length, travel time, and also route popularity, which are enabled by common driving behaviors found in available trajectory data.**

The ability to evaluate the quality of routing services enables service providers to improve the quality of their services and enables users to identify the services that best serve their needs. The paper covers experiments with real vehicle trajectory data and an existing online navigation service. It is found that the availability of information about previous trips enables better prediction of route travel time and makes it possible to provide the users with more popular routes than does a conventional navigation service.

I. Introduction

Traveling is part of most peoples' daily life, and each day we travel to different places for different reasons. Car drivers often possess a good knowledge of their local area and tend to follow familiar routes to different destinations. Depending on specific driving habits, local drivers may follow routes with few turns or with few traffic lights or speed bumps, and they may have special shortcuts. Such preferred routes are not necessarily the shortest or the fastest.

When we travel to unknown destinations, we often depend on an available navigation service. In particular, when we plan a trip, most probably, we would like to use the best route possible. Depending on our preferences, the best route can be the fastest or the shortest route, one that is mostly constrained to the main roads, etc. By entering start and end locations, with only one click, we receive all the information needed to proceed with the trip. As users, we simply rely on the provided route, and we often do not consider possibly better alternatives.

Consider three different routes between start location S and destination D, as shown in Figure 1. Route $\sharp 1$ (dotted line) covers the longest segment of the road with speed limit 70, and this route is the fastest, since all other roads have speed limit 50. Route $\sharp 2$ (solid line) is the shortest route. It goes straight to the destination and does not include any detours. Next, local drivers prefer Route $\sharp 3$ (dashed line). It can be seen that Route $\sharp 1$ passes by *Food Store* and its parking lot. This

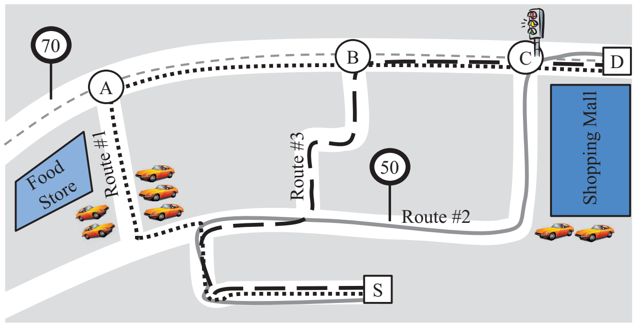

Fig. 1. Example of Trips

often causes inconveniences due to cars entering and leaving the lot. Route $\sharp 2$ passes by *Shopping Mall* and intersection C, which is controlled by traffic lights. When taking this route, it is easy to get stuck in a traffic jam, especially during rush hour. Thus, Route $\sharp 3$ is the preferred one, and navigation services can benefit from knowledge of the habits of local drivers.

Several techniques have been introduced that aim to improve the quality of routing services by using drivers' trajectories [5], [7], [21]. The main idea of these proposals is to form a road network from the road segments that are covered by trajectory data set. A resulting route is formed by prioritizing parts of roads that are taken mostly by the specific driver [5] or by other drivers [7], [21].

Thus, a suggested route is the most favorite or the most popular route in the available data set. To evaluate the quality of proposed routes, Ying et al. [21] compare their time consumption with the time consumption of the ones provided by Google Maps. Since it is possible that routing services are already providing the same route, this time-focused comparison does not provide any insight into how good routing services are at suggesting the preferred routes. Not only the time consumption is important, but also the route itself.

We present a framework that enables the comparison of the routes provided by routing services with the travel behavior of local drivers. The framework relies on the availability of vehicle trajectory data for its operations. As such data becomes available in increasing volumes, the framework gains in applicability and utility. The framework has three main modules: GPS data preprocessing, route identification, and route comparison.

The preprocessing module employs a set of methods to clean the GPS data and to identify trajectories. These trajectories are grouped such that the trajectories in a group are between the same pair of a start and a destination location. In the route identification module, trajectories that follow the same route are identified. Comparison between trajectories is done using a similarity definition that is based on the concept of Longest Common Subsequence (LCSS) [19]. The route comparison module implements a set of experiments that compare the routes of local drivers against the routes provided by a navigation service. Routes are compared according to three metrics: route length, route travel time, and route popularity.

Example findings from the empirical study include:

- Drivers follow multiple routes between source and destination locations, and 61% of these pairs have a route that matches the one provided by the navigation service considered.
- When drivers follow a route provided by the navigation service, that route is typically the most frequently used among the alternatives.
- Routes that are more popular among drivers than the ones provided by the navigation service are not necessarily shorter or take less time.
- The travel time predicted by the navigation service varies on average $\pm 20\%$ from the average time used by drivers.

The paper's objective is to provide data-based insight into routing service quality, rather than to provide new, scalable GPS data processing algorithms. The ability to evaluate the quality of a navigation service brings benefits to both service providers and users. Service providers can evaluate the quality of their services and, if needed, improve on them, e.g., by taking into account information about previously taken trips. For example, route recommendations to a driver ca be based on the driving habits of other drivers with similar demographic profile. Users can determine which routing services best meet their needs and can choose a service to use on this basis.

Our study can also enable the creation of a *meta*-routing service that makes use of several preexisting navigation services. Together with the trajectory data, comparisons of results from different services can be made, and the user can be provided with the route that best reflects local driver behavior.

In summary, the main contributions of the paper are:

- We propose a framework with concepts and techniques that enables comparison of routes collected using GPS device and routes provided by a navigation service.
- We report findings from a study with real data from 144 drivers covering a period of four months that provides insight into travel behavior of local drivers, i.e., route diversity and popularity.
- We evaluate the performance of a popular navigation service using three metrics: route length, travel time, and route popularity.
- We provide empirical evidence that the use of travel histories of drivers can increase the quality of a navigation service by providing users with more popular routes and make travel time predictions more accurate.

The remainder of the paper is organized as follows. Section II provides the main definitions and presents the proposed framework. Section III reports on the results of the empirical study. Related work is covered in Section IV. Section V concludes the paper.

II. Routing Service Quality Evaluation Framework

We proceed to define key concepts used in the paper. Then we describe proposed framework that enables the comparison of the routes provided by routing services with the actual driving behaviors of local drivers.

A. Definitions

A GPS log is a collection of GPS points $pt = (p, t)$, where $p = (x, y)$ is a point location in two-dimensional Euclidean space $\mathbb{R}^2$ and t is a timestamp. Such a log captures the trajectory of a moving object during a monitoring period. We consider logs obtained from users that are driving vehicles. Locations where users stay for longer than certain duration of time, the stay duration threshold, are called *stay points*. Each stay point can be either at the beginning or at the end of a trip.

We consolidate stay points that are near each other by grouping them into a location object, which is a circular region. To avoid location objects with large radiuses, start and end stay points are assigned to separate location objects.

Definition 1: A **location object** lo is defined as a triple (c, tp, P). Here $c = (p_0, rd) \in \mathbb{R}^2 \times \mathbb{R}$ denotes the circle with center p_0 and radius rd; $tp \in \{1, 0\}$ captures the object type, i.e., begin or end, respectively; and P is a set of stay points. The center p_0 of the circle is the mass center of the set of stay points in P. We denote the set of all location objects by LO.

A trip made by a user from one stay point to another is called a *traversal* and is defined as follows.

Definition 2: A **traversal** tr is a pair (u, pl), where $u \in U$ identifies a user and $pl = \langle pt_1, \ldots, pt_N \rangle$ 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 $t_i < t_{i+1}$, $i = 1, \ldots, N$. The set of all traversals is TR.

The polyline in a traversal is the traversal's trajectory. The number of points in a trajectory depends on the sampling frequency of the GPS device used, the duration of the monitoring period, and the stay duration threshold.

A traversal starts and ends at particular start and end location objects. Traversals that share the same pair of location objects are identified by the corresponding *source–destination* pair.

Definition 3: A **source–destination pair** sdp is a pair (lo_s, lo_d) of two location objects lo_s and lo_d. We use SDP to denote the set of all source–destination pairs.

We use dot notation to reference the components of a structured element. Thus, two traversals $tr, tr' \in TR$ belong to the same source–destination pair $sdp = (lo_s, lo_d)$ iff $lo_s.tp = 1$ and $lo_s.c$ spatially contains $tr.pt_1.p$ and $tr'.pt_1.p$

and $lo_d.tp = 0$ and $lo_d.c$ spatially contains $tr.pt_N.p$ and $tr'.pt_N.p$.

In our setting, it is helpful to be able to collect all geographically similar traversals for a source—destination pair into a single structure, which we call a *route object*.

Definition 4: A **route object** r is defined as a four-tuple $(sdp, \{u_i\}, \{pl_i\}, time_{avg})$, where sdp is a source–destination pair, the u_i are users, the pl_i are trajectory polylines, and $time_{avg} = (\sum_{i=1}^{N} pl_i.pt_{N^i}.t - pl_i.pt_1.t))/N$ is the average time needed to traverse a polyline in the route object. Each user and each polyline in a routes object stem from some traversal that belongs to the object's source–destination pair. The set R denotes the set of all possible routes objects.

Example 1: Figure 2 illustrates three traversals by the users u_1, u_2, and u_3. The polylines of the traversals are given by pl_1, pl_2, pl_3. The figure shows their projections into $\mathbb{R}^2$. Polyline pl_1 is formed from a sequence of six points: $pl_1 = \langle(p_1, t_1), (p_2, t_2), \ldots, (p_6, t_6)\rangle$. Begin and end location objects are formed from the stay points of the three traversals: $A = (c_1, 1, \{p_1, p_7, p_9\})$, $B = (c_2, 0, \{p_6, p_8, p_{10}\})$, where $c_1 = (p_7, rd_1)$ and $c_2 = (p_8, rd_2)$.

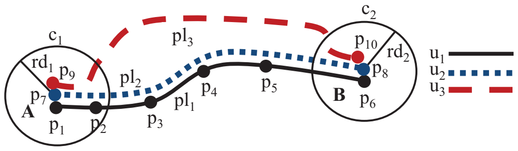

Fig. 2. Example of Trips

Location objects A and B are combined into the *source–destination* pair $sdp = (A, B)$. Three traversals belong to sdp, e.g., the traversal $tr_1 = (u_1, pl_1)$. Two routes objects are formed: $r_1 = (sdp, \{u_1, u_2\}, \{pl_1, pl_2\}, time_{avg_1})$ and $r_2 = (sdp, \{u_3\}, \{pl_3\}, time_{avg_2})$. $\square$

There can be several different routes between a source–destination pair. To be able to identify the route that is the most preferred among users, we define a preference function.

Definition 5: (**Preference Function**) Given a route object $r = (sdp, \{u_i\}, \{pl_i\}, time_{avg}) \in R$, the score of a route is defined as follows:

$$\text{score}(r) = \alpha|r.U| + (1-\alpha)\,|r.PL|,$$

where $0 \leq \alpha \leq 1$ is user-provided parameter that can balance the importance of the number of different users taking the route and the number of traversals.

We define the *preferred route* pr for a source-destination pair sdp' as the route object for that pair with the highest score:

$$pr = \underset{r \in R, r.sdp = sdp'}{\text{argmax}} \quad (\text{score}(r))$$

Example 2: Continuing with the example from Figure 2, let $\alpha = 0.5$, meaning that the number of drivers taking a route and the number of traversals are equally important in calculating the score of the route. Then $\text{score}(r_1) = 0.5 \cdot 2 + (1-0.5) \cdot 2 = 2$ and $\text{score}(r_2) = 0.5 \cdot 1 + (1-0.5) \cdot 1 = 1$, which means that route r_1 is the preferred route between the source–destination pair $sdp = (A, B)$.

B. Framework

The framework for routing service quality evaluation encompasses the system whose architecture is described in Figure 3. It consists of four main data processing components. All computation is done in an offline mode.

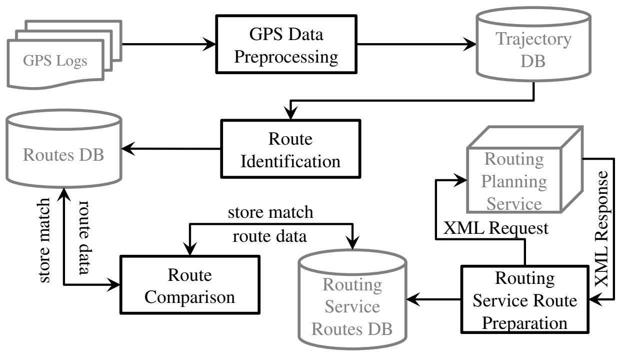

Fig. 3. Framework Overview

1) GPS Data Preprocessing: In this step, GPS data is processed to identify separate traversals of different routes based on GPS log data from local drivers.

Trajectory Segmentation: We identify trajectories, each of which represents one route traversal. The GPS log contains GPS points obtained from the cars during the monitoring period. To distinguish the different trips made by a driver in a car, the log of GPS points from each car was partitioned when duration between the timestamps of consecutive GPS points exceeded a stay duration threshold of $maxT$ sec. In addition, outlier GPS points caused by malfunctioning of the GPS device were removed. Depending on the specifics of the GPS logs used, additional steps might be included, e.g., the identification of *stay points* [23].

Trajectory Filtering: In our setting, GPS points were generally obtained from a car when the car's engine was on. Due to the different driving-related habits of different individuals, some of the traversals initially identified based on the GPS points are of little use in our study. To achieve a higher-quality evaluation, we thus filter the initial set of traversals.

First, we remove traversals that are shorter than $minLen$ m and have a time duration of less than $minT$ sec. Next, we remove traversals that have $||pl.pt_1.p, pl.pt_N.p|| < minDist$ m, where $||\cdot,\cdot||$ denotes Euclidean distance. This eliminates traversals where a driver visits a destination, but does not turn off the car and then returns near to the start location. Finally, we remove traversals that have $\text{length}(pl) \geq \delta \cdot ||pl.pt_1.p, pl.pt_N.p||$, where $\text{length}(\cdot)$ denotes the length of an argument polyline. This filters traversals that involve substantial detours, which can occur for many reasons. For example, a driver on the way to work may pick up a co-worker who lives further away than the place of work.

Detection of Source–Destination Pairs: In this step, we identify meaningful locations for the drivers and the traversals that occur between these locations.

Using Algorithm 1, a set of location objects is identified. Begin and end points of the trips are joined into location objects with separate types (lines 2–6). In lines 4 and 5 location objects are created using Algorithm 2.

In Algorithm 2, if there exists an object of the right type that spatially contains a new point then the new point is added to the point set, and the mass center of the circle is updated (lines 2–4). If such an object does not exist then a new location object of the provided type is created with an initial radius of rad m (lines 5–6).

Algorithm 1 createLocObjs(PL_{in})

```
IN: $PL_{in} \subseteq PL$
OUT: $LO_{out} \subseteq LO$
1: $LO_{out} \leftarrow \emptyset$;
2: for $pl \in PL_{in}$ do
3:   $p_s \leftarrow pl.pt_1.p$; $p_e \leftarrow pl.pt_N.p$;
4:   $LO_{out} \leftarrow \text{addLocObj}(p_s, 1, LO_{out})$;
5:   $LO_{out} \leftarrow \text{addLocObj}(p_e, 0, LO_{out})$;
6: end for
7: for $lo \in LO_{out}$ do
8:   $r_{max} \leftarrow \max(||lo.c.p_0, p_i|| : p_i \in lo.P)$;
9:   $lo.c.r \leftarrow r_{max}$;
10: end for
11: if CheckOverlap($LO_{out}$) = true then
12:   $LO_{out} \leftarrow$ merge overlapping circles;
13: end if
14: return $LO_{out}$;
```

Algorithm 2 addLocObj($p_{in}, type, LO_{in}$)

```
IN: $p_{in} \in P, type \in \mathbb{R}, LO_{in} \subset LO$
OUT: $LO_{out} \subset LO$
1: $lo \leftarrow \bot$;
2: if $\exists lo \in LO_{in}$ : Contains($lo.c, p_{in}$) $\wedge$ $lo.tp = type$ then
3:   $lo.P \leftarrow lo.P \cup p_{in}$;
4:   $lo.c.p_0 \leftarrow$ mass center of $p \in lo.P$;
5: else
6:   $c \leftarrow (p_{in}, rad)$; $lo \leftarrow (c, type)$; $LO_{in} \leftarrow LO_{in} \cup lo$;
7: end if
8: $LO_{out} \leftarrow LO_{in}$;
9: return $LO_{out}$;
```

Continuing with Algorithm 1, after all begin and end points are assigned to location objects, the radiuses are tightened to remove empty space from the circles (lines 7–10). Further, we check whether the regions of the objects overlap and combine intersecting objects into larger ones (lines 11–13). Only one iteration of overlap checking is done. This is to avoid having location objects with overly large radiuses.

Algorithm 3 identifies *source–destination* pairs. First, a set of location objects LO is retrieved using Algorithm 1 (line 1). Then sets of start and end objects are identified (lines 2–3). Finally, location objects are paired into a source–destination pair if at least 3 traversals exist between them (line 4). We take into consideration only such trips because our study focuses on popular routes.

Algorithm 3 identifySourceDestPairs(PL_{in})

```
IN: $PL_{in} \subseteq PL$
OUT: $SDP_{out} \subseteq SDP$
1: $LO \leftarrow \text{createLocObjs}(PL_{in})$;
2: $LO_s \leftarrow \{lo | lo = (c, tp, P) \in LO : tp = 1\}$;
3: $LO_d \leftarrow \{lo | lo = (c, tp, P) \in LO : tp = 0\}$;
4: $SDP_{out} \leftarrow \{sdp \mid sdp = (lo_s, lo_d), lo_s \in LO_s, lo_d \in LO_d, pl_i.pt_1.p \in lo_s.P, pl_i.pt_{N^i}.p \in lo_d.P, pl_i \in PL^* \subseteq PL_{in}, |PL^*| \geq 3\}$;
5: return $SDP_{out}$;
```

2) Routing Service Route Preparation: Algorithm 4 is used to obtain a route from a routing service for each source–destination pair. We use the Google Directions API because this is a popular and state-of-the-art service.

Algorithm 4 getGoogleRoutes(SDP_{in})

```
IN: $SDP_{in} \subseteq SDP$
OUT: $R_{google} \subset R$
1: for $sdp \in SDP_{in}$ do
2:   $p_s \leftarrow sdp.lo_s.c.p_0$; $p_e \leftarrow sdp.lo_d.c.p_0$;
3:   $(pl, time) \leftarrow \text{getRouteFromGoogleAPI}(p_s, p_e)$;
4:   $r_g \leftarrow (sdp, \emptyset, \{pl\}, time)$;
5:   $R_{google} \leftarrow R_{google} \cup r_g$;
6: end for
7: return $R_{google}$;
```

For each source–destination pair, we query the Google Directions API to obtain the route that Google proposes (lines 1–6). The API takes two point locations as arguments. As the start location, we take the center of the source location object, and as the end location, we use the center of destination location object (line 2). Then the API returns a polyline and a travel time. The algorithm creates a route object using returned polyline and time (line 4).

3) Route Identification: In this step, distinct routes between source–destination pairs are identified.

Different drivers tend to follow diverse routes between the same start and end locations. In order to be able to determine how many different routes were taken, we define a *similarity* measure between traversals.

The set of traversals between the same *source–destination* pair may have different lengths, and the polylines of different traversals may also have different numbers of sampling points. Therefore, a measure based on Euclidean distance is not useful for computing the similarity between traversals. Instead, we exploit the notion of Longest Common Subsequence (LCSS) [19], [20] to define the similarity between two traversals that are represented by sequences of coordinates.

We project the polyline of a traversal into $\mathbb{R}^2$ by omitting its timestamps. 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 $LCSS_\epsilon(pl, pl')$ between polylines pl and pl' is defined as follows, where pl_i is pl constrained to its first i points.

$$LCSS_\epsilon(pl, pl') = \begin{cases} 0 \text{ if } n = 0 \vee m = 0 \\ LCSS_\epsilon(pl_{i-1}, pl'_{j-1}) + 1 \text{ if } |p_i.x - p'_j.x| < \epsilon \\ \qquad\qquad \wedge |p_i.y - p'_j.y| < \epsilon \\ \max\{LCSS_\epsilon(pl_i, pl'_{j-1}), LCSS_\epsilon(pl_{i-1}, pl'_j)\} \text{ otherwise} \end{cases}$$

We then define the similarity S_1 between two trajectories pl and pl' with a given integer value ϵ, as follows:

$$S_1(pl, pl', \epsilon) = \frac{\text{length}(LCSS_\epsilon(pl, pl'))}{\text{length}(pl)} \cdot 100\%$$

Here, $\text{length}(LCSS_\epsilon(pl, pl'))$ returns the length of the polyline that forms the longest common subsequence.

The match between a pair of traversals thus depends on the length of the traversal. If a trip is long, the similarity has to be higher.

We then use function S_1 to identify separate routes for a source–destination pair. Whether two polylines pl and pl' represent separate routes is decided by the following predicate that uses Algorithm 5:

$$C_1(pl, pl', \epsilon) = \begin{cases} true & \text{if } S_1(pl, pl', \epsilon) \geq \text{getMatch}(pl, pl') \\ false & \text{otherwise} \end{cases}$$

Algorithm 5 getMatch(pl, pl')

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

Two parameters, $sim_{min} \in \mathbb{N}$ and $sim_{max} \in \mathbb{N}$, are used to calculate the similarity value to be satisfied. The possible difference between two polylines is in the range $[sim_{min}, sim_{max}]$ and takes into account the average length of polylines pl and pl' (line 2).

Example 3: Figure 4 depicts two trajectories $pl = \langle p_1, \ldots, p_{11}\rangle$ and $pl' = \langle p'_1, \ldots, p'_{14}\rangle$. The gray buffer around trajectory pl exemplifies the threshold ϵ. Let $\text{length}(pl) = 104$, $\text{length}(pl') = 110$, $sim_{min} = 1$, and $sim_{max} = 10$.

Then $\text{length}(LCSS_\epsilon(pl,pl')) = \text{length}(\langle p_1, p_2, \ldots, p_7\rangle) + \text{length}(\langle p_{10}, p_{11}\rangle) = 61 + 12 = 73$ is the length of the LCSS trajectory. To obtain a match between pl and pl', we require that getMatch returns 99%. The similarity function S_1 gives $S_1(pl, pl', \epsilon) = \frac{73}{104} \cdot 100\% = 70\%$. Therefore, the two polylines are considered as traversals of different routes: $C_1(pl', pl'', \epsilon) = false$. □

4) Route Comparison: In the last step, we determine the similarity between routes obtained from the GPS logs and routes obtained from the Google Directions API.

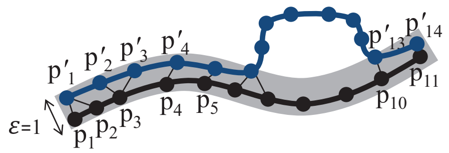

Fig. 4. Comparison of Trajectories

Drivers often start their trips from a parking lot, which is not on the road, while Google routes start from road locations. Therefore, to obtain a more accurate comparison of the length and time consumption of routes, an additional preprocessing step is performed: *polyline trimming*.

All the polylines between source–destination pairs are trimmed using Algorithm 6. The idea is to remove the beginnings and endings from the GPS polylines, so that their starts and ends are closer to the first and last points in the Google routes.

Algorithm 6 trimPolylines(R_g, R_d)

```
IN: $R_g, R_d \subseteq R$
OUT: $R_d$
1: for $r_g \in R_g$ do
2:   $pl_g \leftarrow r_g.pl$;
3:   for $r_d \in R_d : r_d.sdp = r_g.sdp$ do
4:     $PL' \leftarrow \emptyset$;
5:     for $pl \in r_d.PL_r$ do
6:       $pl' \leftarrow \text{subpolyline}(pl, pl_g.pt_1, pl_g.pt_N)$;
7:       $PL' \leftarrow PL' \cup pl'$;
8:     end for
9:     $r_d.PL_r \leftarrow PL'$;
10:  end for
11: end for
12: return $R_d$;
```

The algorithm takes a set of GPS routes and a set of Google routes as input and returns a set of GPS routes with updated polylines. For each Google route, a set of GPS routes with the same source–destination pair is selected (lines 1–3). First, a set of new polylines is assigned to an empty set (line 4). Then, for each old polyline of the route, new trimmed polylines are computed using function $\text{subpolyline}(pl, pl_g.pt_1, pl_g.pt_N)$ (lines 5–8). Finally, a new set of polylines is assigned to the route object (line 9).

To complete the framework, we define the notion of a match between a GPS route and a Google route. We again use LCSS-based similarity, which allows gaps between sequences and is able to handle situations where one of the traversals is represented by a more sparse polyline than is the other traversal.

Definition 6: Let r_d and r_g be a GPS route and a navigation service route, respectively. A match occurs between r_d and r_g ($r_d \equiv r_g$), when:

1) r_d and r_g have the same start–destination pair and
2) $\exists pl \in r_d.PL_r : C_1(pl, pl_g) = true$, where $pl_g \in r_g.PL_r$.

III. Empirical Analysis of Route Quality

This section presents the evaluation of route quality. Evaluation is done by comparing routes proposed by a routing service against routes taken by local drivers.

A. Data Description

We use high-quality data collected at 1 Hz from drivers in the "Pay as You Speed" project[1]. The project took place in North Jutland, Denmark and was conducted by the Traffic Research Group at Aalborg University. In the project, young drivers who took part in an intelligent speed adaptation experiment were tracked. This data is well suited for our study because comparison of traversals can be done using the sampled GPS points directly. In contrast, the use of data sampled at a low rate calls for additional preprocessing, including complex map matching [22].

Parameter settings that were found to yield meaningful results in our study are given in Table I.

TABLE I
Experimental Settings

Parameter	Value	Parameter	Value
$maxT$ (sec)	180	$minLen$ (m)	500
$minDur$ (sec)	180	$minDist$ (m)	500
δ	3	rad (m)	20
ϵ	50	$[sim_{min}, sim_{max}]$ (%)	[1,10]

The subset of the entire data set that we use contains 53,129,702 GPS points from 144 cars and drivers during a monitoring period that extends from October 1, 2007 to January 31, 2008. The numbers of traversals after each preprocessing step are given in Table II.

TABLE II
GPS Data Preprocessing Results

Preprocessing Step	# Traversals
Trajectory segmentation	~71,700
Trajectory filtering	53,638
Source–destination pairs	24,735

The final dataset contains 24,735 traversals involving 1,374 source and 1,277 destination objects that have average radiuses of 18 and 16 m, respectively. From this set of traversals, we identified 2,910 source–destination pairs with 4,950 distinct routes.

The GPS routes were evaluated against the top-1 result provided by the Google Directions API[2].

Figure 5 show distribution of routes and traversals across different route lengths. Most routes are shorter than 20 km, and 2% of all routes exceed 50 km. Similar observations hold for traversals.

Figure 6 presents the distribution of distinct routes between source–destination pairs across average route length. For all average route length intervals, the minimum number of distinct routes for a source–destination pair is one, while the highest number of distinct routes for a source—destination pair is 12.

[1]http://www.sparpaafarten.dk

[2]https://developers.google.com/maps/documentation/directions/

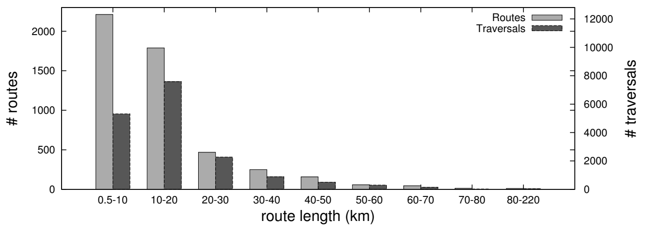

Fig. 5. Distribution of Routes and Traversals

On average, a source–destination pair has between one and two routes. Thus, a source–destination pair typically has only few distinct routes taken by local drivers. Routes with average length above 80 km form only a small part of our dataset and are not shown.

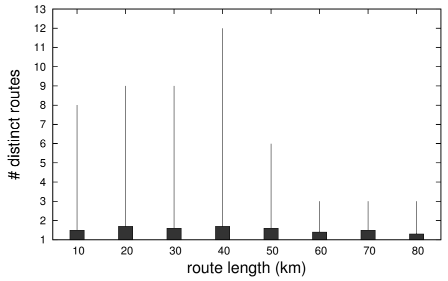

Fig. 6. Route Diversity

Figure 7 shows how often on average the identified routes were traversed across different route lengths. We see that one distinct route was taken at most 6 and at least 2 times, on average.

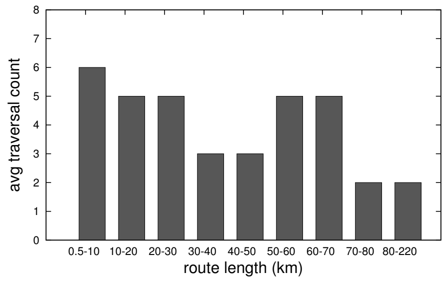

Fig. 7. Average Usage

B. Local Driver Routes vs. Routing Service Routes

We aim to compare routes taken by local drivers against the routes that are provided by the online routing service.

As explained earlier, our cleaned dataset contains 2910 source–destination pairs. Using Definition 6 (match between routes), 61% of these pairs have at least one route that matches the route provided by the Google Directions API.

Figure 8 shows the percentage of source–destination pairs that have a matching Google route for different average route lengths. The largest percentages of matches are found for the shorter routes. For example, for routes between 0.5 and 10 km, about 73% matches exist. Routes between 10 and 20 km have matching traversals in 56% of the cases. The percentages of matches decrease with increasing route length. This might be

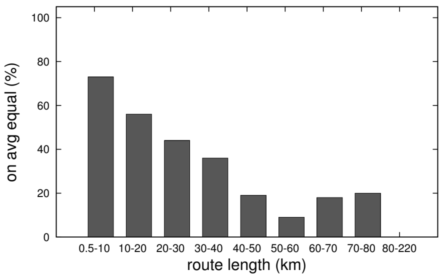

Fig. 8. Matching of Driver Routes to Google Routes

due to an increase in the possible routes between sources and destinations that are further apart. The increase in matches for routes with length between 60 and 80 km may simply be due to the fact that the dataset has few distinct routes in this length range.

Even if a GPS route has a matching Google route, the GPS route may not be popular. Thus, we investigate the numbers of traversals of GPS routes that match Google routes. For example, assume one source–destination pair has 10 traversals and 2 of them have a matching Google route. Then, 20% of the traversals correspond to the route provided by the Google Directions API. The results presented in Figure 9 show that Google routes are quite popular, accounting for more than 60% of all matching traversals.

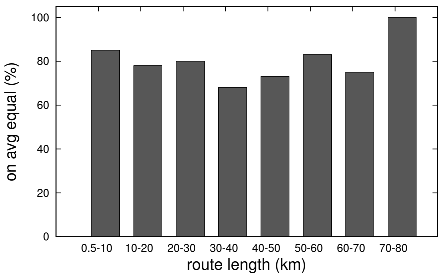

Fig. 9. Traversals With a Matching Google Route

Now let us consider how many Google routes are *preferred routes* for source–destination pairs in the dataset. When calculating the score of each route we used $\alpha = 0.5$, which assigns the same weight to the number of different drivers taking a route and the number of traversals of the route. The results are shown in Figure 10.

Most source–destination pairs in our dataset concern routes that are up to 20 km long. Here we see that the Google routes coincide with the route preferred by drivers for 73% of the source–destination pairs for routes up to 10 km, while the Google routes are the preferred routes for 56% of the source–destination pairs with length 10 to 20 km. As the number of routes available in our dataset decreases, the percentage of Google routes that are preferred also decreases.

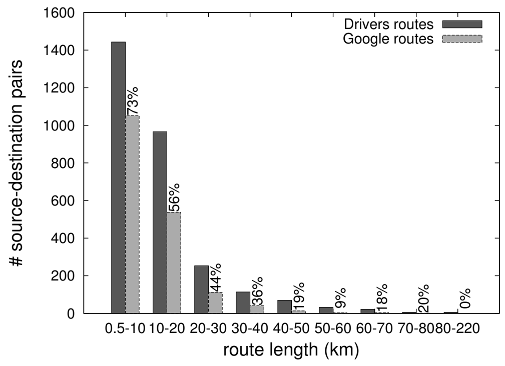

Fig. 10. Preferred Routes

C. Analysis of Routes Taken by Multiple Drivers

Recall that we have trips made by 144 drivers during a 4-month period. We proceed to investigate the selection of routes depending on how many different drivers share a route. Among the 4,950 routes in our dataset, 553 were taken by two or more drivers. Note that several routes can be shared among multiple drivers for one source–destination pair. As shown in Figure 11, the number of drivers taking the same shared route varies from 2 to 9. A total of 334 routes are shared by two drivers, and 55% of these differ from the route suggested by Google.

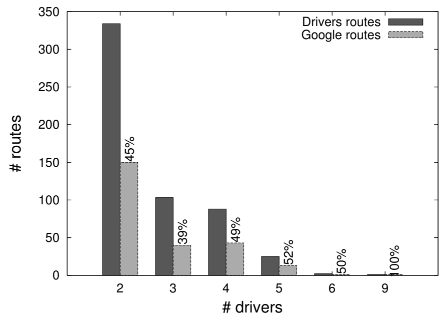

Fig. 11. Analysis of Shared Routes

Our data set contains 248 source–destination pairs that have only one distinct route that is taken by multiple drivers. An analysis of this data is shown in Figure 12. When comparing to the percentages in Figure 11, we see higher percentages for routes taken by 2, 3, and 4 drivers, 61%, 51%, and 57%, respectively. The rest of the percentages stay almost the same.

D. Analysis of Travel Time

We proceed to study route travel time. First, we compare the travel times provided by the Google Directions API with the travel times of corresponding route traversals found in our dataset. Second, we investigate the time consumption and lengths of routes for routes that are preferred by drivers, but are different from the routes provided by Google.

Out of the 2,910 source–destination pairs in our dataset, 1,764 pairs have a route that matches the route provided by Google for the pair. Among these 1,764 routes, 308 take

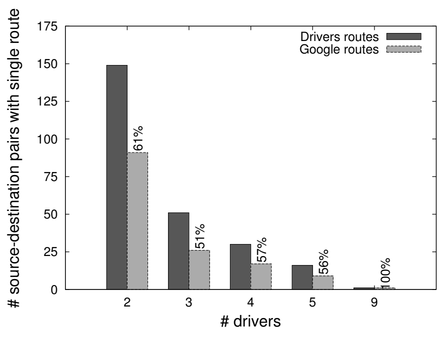

Fig. 12. Multiple Drivers For a Single *sdp* Route

longer to traverse than the time provided by Google. In Figure 13, bars show the distribution of routes across different route lengths. The numbers at the top of each bar state the

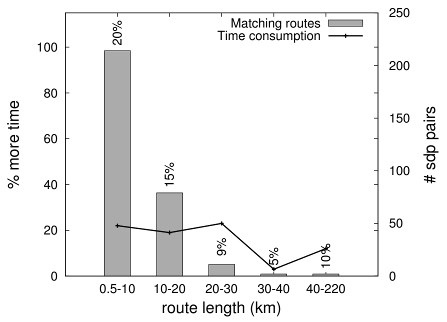

Fig. 13. Matching Routes That Take Longer Time

percentage of the matching routes of the given length that take more time. It is shown that shorter trips take up to 19-23% longer time than predicted.

Next, Figure 14 considers the time consumption for routes that take less time than predicted by Google. Our dataset contains 1,417 such routes. Again, bars show the distribution of routes across different lengths. Numbers on top of each bar indicate the percentage of the matching routes of the same length that take less time. On average, routes take 19% less time than predicted. In our data set, 39 routes take the same

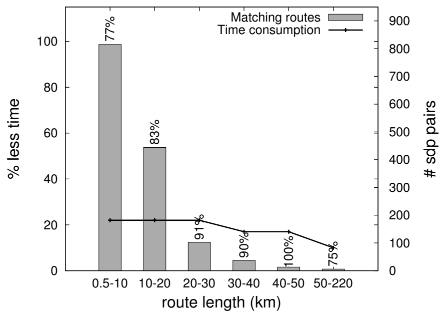

Fig. 14. Matching Routes That Take Less Time

time as predicted by Google.

This experiment shows that most of the matching routes, i.e., 82%, take less time than predicted; and it shows that on average, the predicted times are 20% higher or less than the actual times. Thus, the use of GPS data has a substantial potential for improving travel time estimation.

We also studied the GPS routes between source–destination pairs for which no route matches the Google route. We have 1,146 such pairs. Comparison is done using preferred routes from local drivers. To calculate the score of each route we used $\alpha = 0.75$, to prioritize routes taken by multiple drivers.

First, we present an analysis in relation to route length. Routes that are preferred by drivers are not necessarily shorter. We identified 636 routes, i.e., 55%, that are shorter than the Google routes between the same source–destination pairs.

Results are shown in Figure 15. Bars show the number of source—destination pairs that do not have a route that matches the Google route for the pair. Numbers at the top of each bar give the percentage of such pairs among full dataset. The two curves indicate how much longer or shorter the preferred routes are on average.

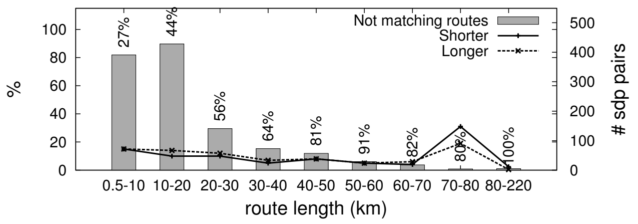

Fig. 15. Analysis of Not Matching Routes Length

It can be observed that the differences in length vary from 4% to 8% for routes between 30 and 70 km long, and from 10% to 15% for routes up to 30 km long. The length range increased considerably for trips between 70 and 80 km long. However, we have only 4 source–destination pairs for this range.

Finally, we consider travel time. We found 873 preferred routes, i.e., 76% of all the routes considered, that take less time than the Google route for the same source–destination pair. Figure 16 shows that on average, the travel times of preferred routes vary from 15% lower to 18% higher.

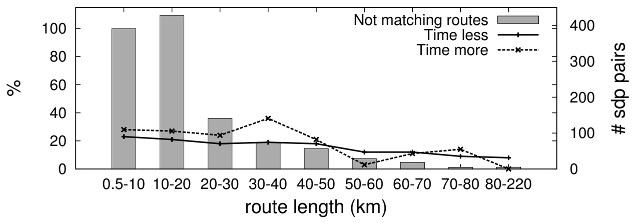

Fig. 16. Travel Times of Routes Without a Matching Google Route

This experiment shows that drivers do not necessarily prefer routes that are shorter or take less time. We can only assume that the many "long" and "slow" preferred routes have other desired properties that are known to local drivers.

IV. Related Work

GPS Data in Routing Services

Some previous studies attempt to increase the quality of routing results by employing knowledge of previous trips. Yuan et al. [21] propose a service that uses taxi trajectory data for route planning. A road network is created from roads that are covered by GPS data. This study also includes a comparison with routes from Google Maps. It is found that when using roads that are often used by taxis, travel time can be reduced by up to 26%. This study only considers travel time; the actual routes taken are not studied.

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. In our study, we do not compute routes. Rather, we focus solely on analyzing routes taken by local drivers and on comparing them with the routes provided by an existing routing service.

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 using *transfer nodes* that are significant location for the users. An edge between two such nodes models the existence of a trajectory that starts at one node and ends at the other. Similarly, when we discover routes taken by local drivers, we form location objects that may be seen as representing locations that are important to the drivers. However, in contrast to this study, we do not need to know how the location objects connect with one another because we analyze routes separately.

Trajectory Representation and Similarity

A driver who has taken the same route multiple times is likely to know the surroundings. Therefore, one part of our study aims to identify popular routes from historical trajectory data. A GPS trajectory is a smooth polyline corresponding to a time-ordered sequence of GPS points. A typical sampling frequency is 1 Hz. In contrast, a trajectory from the Google Directions API consists only of key points along a polyline. The present study calls for means of comparing such differently represented trajectories.

A first step in our framework is to identify separate trajectories in a GPS log. An existing trajectory cleaning framework [10] suggests three preprocessing steps. When an object stays in a location relatively long, a stay point is identified, and the GPS log is split. Then missing segments are interpolated for trips with gaps. Finally, GPS points that have the same coordinates, but different timestamps, or have speed equal to zero are removed, and trajectories with too few points are eliminated.

Zheng et al. [22] provide interesting techniques for the creation of routes from low-frequency GPS data.

Our preprocessing follows steps similar to the above. We split a GPS log into separate trajectories and then filter them. We remove short trajectories, eliminate trips with possible detours, and remove those that have start and end points in close proximity. We use a high-frequency GPS data set that does not call for use of the Zheng et al.'s techniques.

Next, 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 [2], [8], [15] 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) [19], [20] 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 assigning penalties. It is based on edit distance on strings and is suitable for comparison of trajectories with substantial noise. 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.

The second category shape-based similarity measures [1], [11], [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 [13], [14], [16], [17]. 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 [17] or a sequence of edges [13], [14], [16]. 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.

Although we use GPS data received from cars, we chose to not use network-based trajectory similarity. The main reason is the inaccuracy of map-matching algorithms. Roh et al. [14] study map-matching quality but consider only 213 real trajectories. Later work [13] reports on experiments with a larger real data set, but the quality of map-matching is not discussed. Other proposals [16], [17] use synthetic data [3] to evaluate their proposals, rendering map-matching unnecessary.

Trasarti et al. [18] propose methodology for extracting mobility profiles for individual from GPS traces. In order to identify typical behaviors of users from their GPS traces, trips are formed and clustered, with outliers being removed. Then representative trips are extracted. In comparison, we identify routes between sources and destinations, in part to determine how many alternative routes drivers have. While it would also have been possible to extract representative trips for each identified route, we instead compare each traversal to routes provided by the Google Directions API and identify the most similar one.

A final study takes spatial and temporal concepts into account when grouping trips [18]. This was done in order to identify differences in routes taken during the day. In our setting, we consider spatial aspects, since in our study, it does not matter when a trip was taken.

V. Conclusions and Future Work

We present a framework that enables the comparison of routes provided by an available, state-of-the-art routing service with the travel behavior of local drivers. The framework uses vehicle trajectory data formed from GPS tracking logs for its operation. Its techniques and algorithms enable the transformation of GPS data into a format that is suited for the comparison of GPS routes with the differently represented routes provided by a routing service. In addition, it provides means of accomplishing this comparison.

The framework consists of three modules. The preprocessing module provides techniques that enable the cleaning of the GPS data and the identification of trajectories. The route identification module uses a trajectory similarity notion based on Longest Common Subsequence to identify trajectories that follow the same route. The route comparison module provides tools that enable the comparison of the routes taken by local drivers with those provided by available routing services.

We report on a study that compares routes according to three metrics: route distance, travel time, and route popularity. Specifically, we report on experiments that show that the use of travel histories of local drivers as captured by GPS devices hold the potential to significantly increase the quality of existing routing services. The experimental findings show that not only can the travel time be predicted more accurately, but users can also be provided with routes that reflect the behavior of local drivers. In many cases, local drivers prefer routes that are neither the shortest nor the fastest.

The study can be extended to cover also temporal aspects such as daily, weekly, or seasonal patterns in the routes chosen by drivers. This will yield drivers' route preferences.

The study focuses on routes between locations that are important to specific individuals. It may also be of interest to study drivers' behaviors in relation to shorter and more frequent trips, e.g., between major junctions or significant points of interest.

Next, it is an interesting and challenging research direction to attempt to incorporate the techniques in the paper's framework into a navigation service to improve routing quality. One more specific challenge is the use of driver behavior in a feedback loop that enables continuous improvement of routing quality. The development of a system that is able to ingest and use past driving data in route prediction is among our future plans.

Acknowledgments

This research was supported in part by the Geocrowd Initial Training Network, funded by the European Commission as an FP7 Peoples Marie Curie Action under grant agreement number 264994.

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] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[3] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[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. 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] A. Idrissov and M. A. Nascimento. A trajectory cleaning framework for trajectory clustering. In *Mobile Data Challenge (by Nokia) Workshop*, 2012.

[11] B. Lin and J. Su. One way distance: For shape based similarity search of moving object trajectories. *GeoInformatica*, 12(2):117–142, 2008.

[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] G.-P. Roh and S.-W. Hwang. TPM: Supporting pattern matching queries for road-network trajectory data. In *EDBT*, pages 554–557, 2011.

[14] 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.

[15] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: Fast similarity search under the time warping distance. In *PODS*, pages 326–337, 2005.

[16] E. Tiakas, A. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *J. Syst. Software*, 82(5):772–788, 2009.

[17] 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.

[18] R. Trasarti, F. Pinelli, M. Nanni, and F. Giannotti. Mining mobility user profiles for car pooling. In *KDD*, pages 1190–1198, 2011.

[19] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–685, 2002.

[20] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou. An effectiveness study on trajectory similarity measures. In *ADC*, 2012.

[21] 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.

[22] K. Zheng, Y.u Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE*, pages 1144–1155, 2012.

[23] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*, pages 791–800, 2009.