Travel-Time Estimation in Road Networks Using GPS Data

Anders Forum Jensen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7, building E
DK-9220 Aalborg Ø
dphreak@cs.aau.dk

Troels Villy Larsen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7, building E
DK-9220 Aalborg Ø
chucara@cs.aau.dk

PREFACE

This article is based on previous work by the same authors [13]. The trip-based approach in this article is an extended version of the trip-based approach presented in our previous article. When we talk about the trip-based approach, however, we are referring to the extended trip-based approach. Section 1 has undergone minor revisions while Section 2 is mostly untouched, as, to our knowledge, no new material has become available. Sections 3 has been rewritten, but contains the same information. Section 4 is based on the same data warehouse schema with minor revisions, but the section has had major revisions. Section 5 is new material with the exception of the two small paragraphs in Section 5.1. Sections 6, 7, 8, 9 and 10 are completely new. All algorithms and figures are new with the exception of Figure 3.

The paper version of this article contains a CD with source code. There are 368 lines of SQL, 556 lines of PL/SQL and 7631 lines of PHP. See the readme.txt file at the root of the system for an overview of the contents. The code can also be obtained by contacting the authors at above e-mail addresses.

SUMMARY

Billions of dollars are lost in traffic Worldwide every year. Numerous surveys have shown that people waste several hours a week due to congestion in traffic. Being able to predict when congestion occurs ahead of time would enable drivers to choose a different path than originally planned, and could potentially reduce the amount of hours lost in traffic each day. By analyzing historic traffic data, patterns can be found that enable us to foresee tendencies in traffic. Unfortunately, collecting traffic data is expensive and time consuming, as it requires the use of loop detectors, license plate recognition system, or other specialized hard- or software. By using consumer GPS products, we avoid this problem. Unfortunately, these products do not directly show much about traffic, and they are therefore not as accurate as specialized equipment.

Our goal in this project is to develop a method to use consumer GPS products estimate travel times in a road network accurately. We attempt to achieve this by developing two data-based approaches to travel-time estimating. The point-based approach collects and groups GPS observations per road segment and calculates a travel time based on the average speed. The trip-based approach uses the context of a series of connected observations to determine the exact travel-time for an individual car. By splitting the route of the car into smaller pieces, it can use the route to determine travel-times for any road segment that has been traversed by a car. Like the point-based approach, it uses an average on all travel times to find the final estimate.

We discover that the trip-based approach is more accurate than the point-based approach given the same amount of data. However, if the sampling rate of the data is too low, the point-based approach becomes the better option. By collecting data over an extended period of time, we show that it is possible to estimate travel times fairly accurately. In doing so, we hope to provide a tool for traffic planners and analysts, that could potentially help to improve the traffic situation in large cities.

Travel-Time Estimation in Road Networks Using GPS Data

Anders Forum Jensen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7, building E
DK-9220 Aalborg Ø
dphreak@cs.aau.dk

Troels Villy Larsen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7, building E
DK-9220 Aalborg Ø
chucara@cs.aau.dk

ABSTRACT

Large sums of money are lost every year, as working hours are spent waiting in traffic on congested roads. To avoid this loss, several methods for travel-time estimation has been developed. By determining the travel-time for a particular piece of road ahead of time, congested roads can be avoided.

Traditionally, etimating travel times has relied on slow and costly methods such as loop detectors, observations vehicles or automatic vehicle identification. Other approaches rely on simple calculations based on road lengths and permitted speeds. These approaches are not able to predict traffic, and therefore only give an estimate that applies outside rush hours. Using consumer products to gather data, however, the process can become faster and cheaper as GPS receivers become more abundant.

In this article we develop two approaches to travel-time estimation, the point-based approach and the trip-based approach. Using data from two different data sources as a starting point, we have developed a solution that is able to use very basic data, while still utilizing additional information. We introduce a data warehouse for storing GPS data, a road network and additional data such as information about drivers and vehicles.

In our experiments, we show how the two approaches perform in general and compared to each other. Using our trip-based approach, we are able to provide travel-time estimates with an error rate of 0.3% compared to the actual travel times, which is a major improvement over naive travel times and a slight improvement over our point-based approach.

1. INTRODUCTION

Each day, millions of people waste time waiting in car queues to get to or from work, resulting in money loss as work time is wasted. As traffic volumes increase, the need for precise travel-time estimates grow. Speed limit-based travel times are of little use, and even with better travel-time estimates, traffic changes over time and new travel-time estimates are needed.

Figure 1 shows the travel time on a major road in the city of Aalborg during the day. As can be seen in the figure, the travel-time in one direction is stable while the afternoon rush can easily be identified in the other direction. As a comparison, consider the naive travel time, which is 50% inaccurate at best.

Traffic planning and travel-time estimation has traditionally relied on expensive measureing methods such as loop detectors, vehicle identification devices or floating car observers. With the advent of mobile GPS equipment, new possibilities have opened for cheaper travel-time estimation [7, 19, 21, 27].

Based on standard GPS equipment in vehicles such as taxis, cars, and busses, we can obtain a large number of observations containing the positions and speeds of various

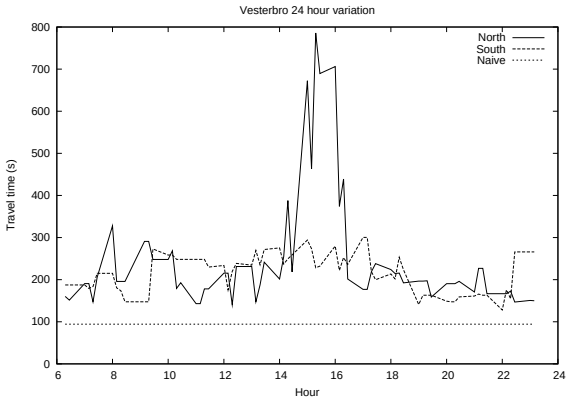

Figure 1: The travel time for Vesterbro during the day.

vehicles. GPS data is inherently imprecise as it is currently impossible to determine the position of a receiver without some error. Obstacles such as overpasses, tunnels, buildings or even trees, can cause gaps in the received data, adding to this inaccuracy. The quality of the GPS data can also vary depending on the GPS device and which parameters are received from the device. Using GPS data from various sources requires special attention to the differences in the data formats. Some logs include information about drivers and vehicles, which can be used to increase performance. For example, slow vehicles can contribute to the estimation is a different way than average cars.

While dedicated moving observer or floating car vehicle-based methods can provide precise estimations, they require that an instructed driver collect the data needed. This is both time consuming and costly as the driver must be paid. This method also provides less data as a relatively small number of vehicles are usually used. Since road networks are ever changing and traffic volumes fluctuate, travel-time estimates must be recalculated occasionally or continually using current data to reflect these changes. For this reason, we consider faster and cheaper methods, such as GPS-based methods, superior to the more precise, but also slow and costly methods.

Previous approaches to travel-time estimation include algorithms based solely on more or less educated guesses calculated from the permitted speed on a particular road segment, on finding a weighted average given single observations, on data collected using expensive moving observer methods, or on the experience of traffic experts [7, 19, 26, 27].

In this article, we propose a method for travel-time es-

timation on road networks that is based primarily on trips found by map matching GPS observations. In some cases data is too sparse to form trips. For these cases we introduce the point-based method, which is a simple average based method, that imposes fewer requirements on the data foundation. The methods used in travel-time estimation in this article are designed solely for planning purposes, as such, dynamic travel-time estimation is not considered in this article. The main contributions of this article are:

- We design a flexible data warehouse scheme to store GPS observations, a road network and related information, e.g. driver and vehicle information.
- We introduce the point-based and trip-based approaches for travel-time estimation.
- We compare the trip-based approach to the point-based approach.
- We study the impact of lowering sampling rates to find a reasonable compromise between precision and data volume.

The rest of this article is structured as follows: In Section 2, we outline existing work. In Section 3, we describe our data foundation, requirements, and prerequisites for travel-time estimation and in Section 4, we introduce our data warehouse. In Sections 5 and 6, we outline our two approaches for travel-time estimation based on GPS data. In Sections 7 and 8, we describe our validation and experiments. Finally, in Section 9, we go through some of the design choices for our system before we conclude and present future work in Section 10.

2. RELATED WORK

Related work in the domain of estimating travel times is focused on three main topics: data collection [12, 19, 22, 24], travel-time and congestion estimation algorithms [4, 7, 19, 21, 22, 23], and storage and retrieval [7, 12, 21]. The first two topics are primarily the concern of industrial research while storage and retrieval is the topic of many academic papers.

The bulk of related work concerned with data collection utilizes a method of collecting data through a limited number of probe vehicles [12, 19, 21, 24], loop detectors [7, 23] or through automated vehicle identification [4]. However, much more data is available if data from all relevant GPS equipped vehicles is utilized. Our aim in this article is to use any GPS data available opposed to using few, specialized, data sources.

Pfoser et al. introduce a dynamic travel-time map based on a spatio-temporal data warehouse in [21]. Floating car data collected using GPS is used to enrich a map with travel times. To maintain coverage on the road network, different fallback methods are proposed and the efficiency of these is studied. The paper also covers important topics such as shortest-path navigation in the map and provides some experiments, although no verification of the developed method is provided. GPS data collected from vehicles not intended for traffic studies is generally classified as moving observer data. The advantage of floating car data over moving observer data is that it is more likely to be accurate since the vehicles are driven according to a strict set of rules. However, floating car data is more expensive than moving observer data, and given enough data, the differences between the results of the two methods disappears. See [24] for a more detailed discussion.

In [12, 19, 21, 24] GPS equipped vehicles are used to collect samples at regular intervals, which are then used for estimating travel times. Many different sampling intervals are used, one minute in [12], 30 seconds in [21] and one second in [24]. Different systems will provide data recorded with different sampling rates, a solution independent of sampling rates has not been proposed to our knowledge. In [22] Quiroga et al. study the impact of changing sampling rate and road segment length using GPS. They show the relationship between sampling rate, segment length, and the resulting errors. They conclude that the sampling rate should not be any longer than six seconds for their specific segment length of about 320 meters.

A number of papers deal with the storage of travel times and road networks using data warehousing or databases in general [2, 10, 14, 15, 21, 25]. In [3], Brilingaite et al. describe a basic map-matching algorithm and present a solution for intelligently enabling routes as context in mobile services.

Güting et al. propose a method for modeling, storing and querying (transportation) networks in databases in [9]. They describe the modeling of routes and junctions based on the conclusion that routes are a better representation for a transportation network than road segments. This perception is shared with (and based on) their earlier work in [16]. In a travel-time application, this assumes that details about road characteristics such as number of lanes and surface quality is available since an entire route cannot otherwise be thought of as a single entity.

Several papers are concerned with actual path finding and travel-time estimation. Kanolus et al. [17] propose a method for finding the fastest path through a road network given the constraints of a time interval at either the start of or destination of the trip. Ku et al. [18] propose an adaptive nearest-neighbour query based on travel time instead of Euclidian or network distance. In [21] Pfoser et al. use the A* algorithm for path finding in a setting much like ours.

In [20], Nielsen presents methods for using data recorded by GPS devices mounted in cars to analyze congestion. It is argued that using GPS data provides more knowledge than traditional methods as routes can be inferred from the stream of GPS observations. In [11], Hansen presents methods for analyzing congestion continually over extended periods of time, using GPS data.

3. DATA FOUNDATION

To be able to accurately estimate travel times, we rely on a number of data sources. We assume that a road center line network is provided that is accurate and complete, including speed limits. For any given road segment, we assume that the permitted speed is constant. We also assume that the direction and speed limit for a road segment does not change during the day. Our map will be stored in the database, and can come in any format, typically a shapefile. We assume that the map is up to date and accurate.

For the GPS data we assume that, as a minimum we are provided with longitude, latitude, a timestamp and either a speed or a unique identification for the recording GPS device. Depending on the GPS device and its user, a number of additional properties may be provided. Depending of which additional properties are provided, different approaches, the point-based or the trip-based, can be applied to the data. If single GPS observations can be linked to each other, either by belonging to the same GPS log or if they are attributed with a unique GPS device identifier, information about the route taken can usually be inferred, depending on sampling rate, speed and road segment length. With this additional route information we can begin to infer which roads where traversed and interpolate timestamps for certain positions along the route. From this route we can create a *trip* which is a list of traversed road segments attributed with times-

tamps.

If additional properties such as vehicle or driver information are supplied, special considerations can be taken. Data recorded by a bus or another heavy vehicle is naturally different from data recorded by a taxi or car. Factors such as acceleration, maximum speed and inaccessible roads mean that data from heavy vehicles is different, but not useless. While the heavy vehicles exhibit other driving patterns than cars or taxis, the data can still be used if special care is taken. We do not have any heavy vehicle data available, and as such we do not make any special measures to ensure that we can use such data. We do, however, have data collected by taxis which may be different from data collected by cars in that taxis stop more often in order to pick up or drop off passengers. Travel-time estimation can be optimized to the vehicle type when information about the vehicle type is available. Similarly, if information about the driver is available, this can also be taken into consideration. Just like vehicle type, drivers may influence the data in specific manners. Young men tend to drive aggressively while the elderly generally drive slower. In general, "the average driver" is assumed to be behind the wheel if no driver information is provided. While we are provided with unique identifiers for some of our drivers, we do not use them in this article.

User behaviour and GPS device precision and quality can lead to degradation in the data quality. For example, GPS devices that do not utilize a magnetic compass but infers heading based on the direction between consecutive points, are often unable to provide a steady heading at low speeds. Furthermore, the precision of the GPS device might not be known or the user might use the device in unexpected ways. We do not assume anything about the quality of the data but use the data as is, with the exception that we filter out data that cannot be matched to the road network or data that is clearly incorrect. For example, observations that are physically impossible or highly unlikely are disregarded. We also disregard observations that cannot be matched to a road segment within 30 meters. This figure is based on an statistical analysis of the data, maximizing the amount of data while removing the least accurate data.

Data sampling rates are another variable. Depending on usage, a GPS device might record data at sub-second precision or every few seconds, minutes or even hours. The data sets available to us are provided by *Bektra* and *Spar på farten*. The *Bektra* data set uses a sampling rate of two minutes which makes it practically impossible to link individual GPS observations to each other. Driver and vehicle information can be linked to the data set using unique identifiers present within the data. The *Spar på farten* data set uses a one second sampling rate and provides a unique GPS device identifier, which makes it possible to link individual GPS observations.

4. DATA WAREHOUSE

In this section, we introduce our data warehouse which is used to store our GPS data and road network.

Figure 2 shows our data warehouse schema. The main table, *gps_fact*, contains the GPS observations, whereas the other tables (dimensions) contain data such as dates, drivers, vehicle types, and road conditions. These tables allow us to apply filtering when querying the fact table. The abbreviations FK and PK are "Foreign Key" and "Primary Key" respectively.

The *gps_fact* table contains information about the location of the observed vehicles (*latitude*, *longitude*, *address_id*), time information (*date_id*, *time_id*), where it is map matched to (*segment_id*, *segment_position*, *precision*), the actual observed data (*speed*, *heading*, *course*) and finally secondary observations (*driver_id*, *vehicle_id*, *source_id*). Also note

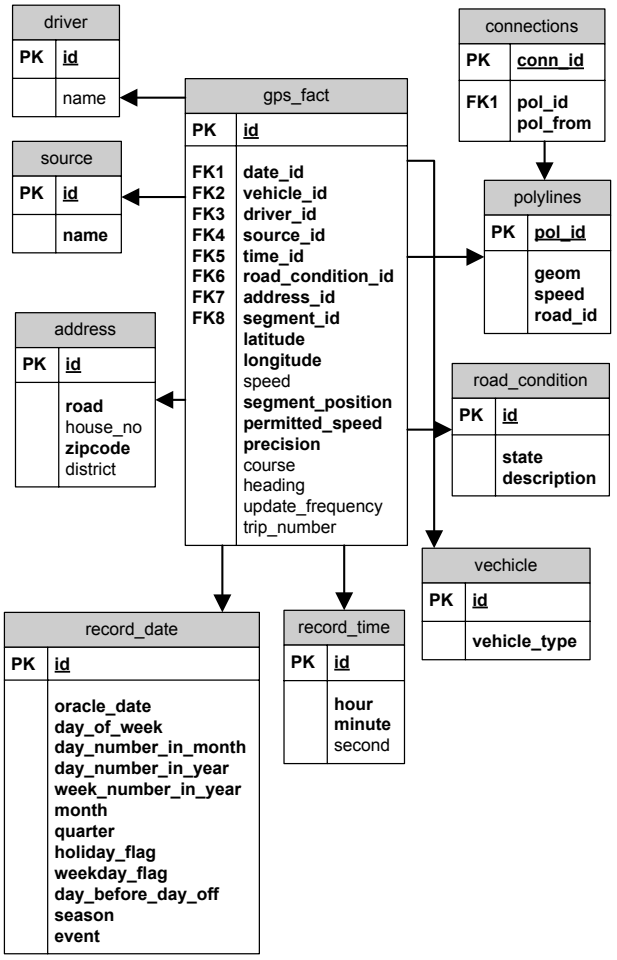

Figure 2: Data warehouse schema

that the speed limit of the road is included in the fact table. This allows us to alter the permitted speed for individual GPS observations, or time periods. This might be useful in cases where the road is subject to changes in speed limit. The table also includes *update_frequency*, which is the number of seconds between each update. This is included to be able to weight the observations differently in order to reduce the impact a single driver can have on the estimation. As an example, imagine 120 cars driving on the same road segment, recording an observation once every 120 seconds. If a single car recording every second drives on the same road segment, that car would have the same impact on the average travel time as the other 120 cars combined.

Heading is derived from course, which is the direction of travel in degrees. If the course of the observation is within 90 degrees of the angle, or direction, of the road segment it has been map matched to i.e. the course runs in the same general direction as the road segment, heading is zero. If it runs opposite the direction of the road heading is one.

The field *trip_number* will be explained in Section 6.2.

The vehicle dimension contains a vehicle type that could provide information about the driving pattern of a particular vehicle. The *vehicle_id* has been set to "Taxi" and "Car" in the *Bektra* data set and *Spar på farten* data set, respectively.

The driver dimension provides us with the means to distinguish between the individual drivers that report to the system. The *driver_id* has been set to the phone number of the driver in the *Bektra* data set while it has been set to "Unknown" in the *Spar på farten* data set, as it is not available. There has not been attributed a specific driving

style to any of the phone numbers.

The time and date dimensions naturally allow us to group the data temporally. The date dimension has fields that allows us to group the data into categories that we expect to have an impact on traffic density, i.e. we would expect traffic to increase the day before a holiday.

The source dimension simply describes the source of the GPS log. In our case, this is *Bektra* or *Spar på farten* as they are currently our only sources.

The address dimension is currently not used and is included only for completeness, but it can be used in cases where the points are map matched to a map that contains, for example, location names or house numbers.

The polylines and connections tables are reused from [3] with only a slight modification: we include a speed limit on each segment. These two tables contain our map as converted from an ESRI Shapefile [5].

5. POINT-BASED APPROACH

The point-based approach views GPS observations as a set of independent points attributed with at least date, time, and speed, which is provided by most GPS devices. The point-based approach calculates a simple, per segment, average travel time and is used, as briefly discussed in Section 3, when individual GPS observations cannot be linked to each other. Given a number of observations on a segment, the point-based travel time is the length of the segment divided by the average speed. An optional parameter is course, which indicates the direction the car is facing. For the rest of this article, we assume the course is provided, as it is a part of the GPS standard. If a course is not provided, we are forced to either dicard the data, or use it for both directions of the road segment.

In practice, the observations would be divided not only by segment but also by e.g. day, time and season. The point-based approach is to prefer when observations are independent, too far apart to unambiguously infer the intermediate route, or when GPS logs without a vehicle or device identifier are composed in such a way that it is not possible to impose an ad-hoc device identifier to interrelated observations. By ad-hoc device identifier, we mean one that can be created based on the context of the data, e.g., if we receive a log file containing a series of observations, each recorded one second later than the preceding. While we do not have a device identifier, we can expect the log to come from the same device and assign the data a generated identifier. Because the point-based approach does not require a vehicle or device identifier, it is also useful in cases, where the anonymity of the driver is important.

5.1 Map Matching

Map matching is relatively easy in the point-based approach because observations are considered to be independent. The simplest possible way of map matching independent points, and the one that we have chosen for the time being, is to do a simple nearest neighbour query. It is possible to enhance the map matching slightly by instead querying for road segments within a certain distance and then, starting with the nearest segment, comparing the course of the observation with the angle of the segment. When GPS observations are independent, map matching can only be as precise as the data, as the two following problems illustrate.

Intersection Matching. A problem that arises when using independent observations is the problem of receiving observations near or at an intersection. Because of the inherent GPS inaccuracy, it is hard to determine which of the possible road segments the observation should be attributed to. Figure 3 shows the problem. Because of the GPS inaccuracy, the small white dot could in fact be anywhere within the bigger red circle. In this case, determining which of the four road segments within the red circle the dot belongs to is problematic. Mapping the observation to a wrong road segment should be avoided if possible as this imposes an incorrect data foundation, which leads to imprecise travel times.

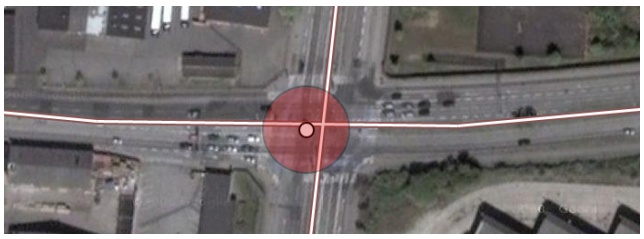

Figure 3: The intersection-matching problem.

A simple way to solve this problem would be to remove all observations that are ambiguous, meaning that they have several possible segments to which they can be matched. We do not consider this an option, however, as we can expect observations near intersections to average to a lower speed than the rest of the road segment. Simply removing them would therefore yield an overly optimistic average speed for the entire segment.

Another solution, which is the one we have chosen, is to assign the observation to the single closest road segment. There is a chance that the observation will be matched to the wrong road segment, but the effect of this error is reduced by the fact that observations are within 30 metres of their actual position 95% of the time. That is, observations are more likely to be accurate, than very inaccurate as observations within 30-300 metres of their actual position only account for 5% of the time [8].

Course Confusion. When stopped at an intersection, some GPS receivers become confused about the course, that is, the direction of travel. The problem arises in devices that are not equipped with a magnetic compass to determine the course. This is also due to the inherent inaccuracy of GPS. Since the course is calculated by analysing the vector created by the last recorded and the current point, the course begins to "jump" from 0 to 360 degrees when stopped at an intersection as the received points scatter around the actual position of the GPS device.

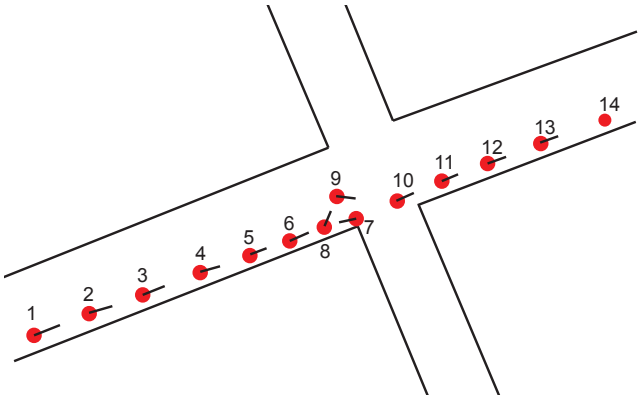

Figure 4: Illustration of changing course at an intersection.

Figure 4 illustrates the problem. The numbers besides the observations (the red dots) indicate the receiving order, and the lines indicate the course. As the car approaches

the intersection, the speed is decreased and at observation 7 the car has stopped. Because of the GPS error, observation 7 to 9 are attributed with a wrong course as described above. Some GPS receivers solve this problem by fixating the course when the speed drops below a certain value. In our example this would mean setting the course of observations 7 to 9 to the value of the course at observation 6. We can emulate this behaviour if the context enables us to identify observations from the same GPS receiver and the sampling rate is sufficient. However, we do not do this as our point-based approach assumes that each observation is independent.

5.2 Storage

While it would be obvious to query the *gps_fact* table during travel-time estimation, the execution time of such a query would increase as more points were entered into the system. In order to still be able to maintain a reasonable response time, we have chosen to create a new table, storing only the travel times. This means that data could be loaded over night, and travel times calculated when execution time is not essential.

Figure 5 shows the table, *point_travel_time*, used to store travel times for the point-based approach. As this approach stores travel times per segment, only two travel times must be stored for each segment per time period, once for each direction of the road segment.

point_travel_time	
PK	**id**
FK1	**segment_id** **travel_time** **heading** **<interval>** count

Figure 5: Table for storing point-based travel times.

In the table *point_travel_time*, the field *travel_time* is used to store the travel time for the road segment in *segment_id*. The field *heading* stores the heading, 0 or 1. The field <interval> represents a compound time interval which can be designed to fit the wanted time granularity. As an example we could store a travel time for each hour in the day, for each day in the week. The <interval> for this would require two fields, *hour* and *day*.

5.3 Calculating Travel Times

Calculating travel times for the point-based approach is a relatively simple task. The process is to calculate a travel time for each road segment in the network at a given time granularity, for example once per 15 minutes of each day. Time periods of similar travel times can be clustered together as discussed in [13].

Algorithm 1 calculates the simple average used in the point-based approach. The algorithm takes two inputs. The first input, R, is a list of the road segments that the travel times should be calculated for. The second input, I, is a list of compound time intervals. Using the example from the previous section, I would contain 168 intervals, 24 for each of the seven days in the week. In the algorithm, on line 4, two different travel times are calculated, one for $h = 0$ and one for $h = 1$. The variable h denotes the heading of the observations used which is either 0 or 1. On line 6 the average for the given road segment r in the time interval <interval> using observations with heading h is found. The

Algorithm 1: Point-Based Algorithm

```
   input  : A list of road segments, R
   input  : A list of intervals, I
 1 begin
 2   avg = 0, len = 0, tt = 0;
 3   foreach r in R do
 4     foreach h in 0, 1 do
 5       foreach <interval> in I do
 6         avg = getAvg(r, h, <interval>);
 7         len = getLength(r);
 8         tt = (len/avg)*3.6;
 9         store(tt, r, h, <interval>);
10 end
```

function *getAvg* can be expressed with this SQL statement:

```
SELECT avg(speed)
FROM gps_fact
WHERE segment_id=r
  AND heading=h
  AND <interval>;
```

Here, <interval> represents the appropriate statement to limit the selection the selected time interval. On line 7 the length of the road segment is found using the following SQL statement:

```
SELECT sdo_geom.sdo_length(p.geom, m.diminfo)
FROM polylines p, user_sdo_geom_metadata m
WHERE p.pol_id=r;
```

In the SQL statement, *sdo_geom.sdo_length()* is a function from the SDO_GEOM package in Oracle Spatial that returns the length of a geometric object. On line 8 the travel time is calculated using the average speed and the length of the road segment. On line 9 the calculated travel-time is stored in table *point_travel_time* described in Section 5.2.

6. TRIP-BASED APPROACH

As briefly discussed in Section 3, trips are a list of traversed road segments along with timestamps denoting when the transition from one road segment to the next occurred. We are able to form trips only when we can determine that a series of observations were recorded by the same GPS device and that those observations are not too far apart. In the *Bektra* data set the sampling rate is 120 seconds, which is far too low to be able to infer routes between the observations. For this article, a trip is defined as follows:

DEFINITION 1. *A trip is defined as a chronologically sorted sequence of observations from the same GPS-receiver unit, each observation having been recorded at no more than a specific number of seconds after the previous observation.*

In this article, the maximum number of seconds between each observation is set to 10. This number is based on an analysis of our data, which is recorded once every second (see Section 8.2). An interval of 10 seconds is large enough to allow gaps in the data due to passing through a tunnel or under a bridge, but is short enough to split trips where the recorder has been turned off for a short period.

The trip-based approach uses a from-to style for calculating travel times as opposed to the *per segment* travel time used in the point-based approach. This means that at an intersection, going right, left or straight through it are stored as three different travel times. The situation is illustrated in Figure 6, where three different destinations are possible from the single originating road segment. This is an improvement over the point-based approach, which was made due to the fact that the time it takes to make a right turn might not

be the same as driving straight through an intersection. As an example, we use one of the larger intersections in the city of Aalborg. Here, a left turn takes 39 seconds, while a right turn takes 19 seconds. Going straight through it takes 20 seconds on average. This illustrates how different directions can have different travel times. This can be due to both the volume of traffic and the properties of the intersection such as turning lanes and light timing. The above example is not the average case, but it does illustate that the travel time for a segment can indeed vary depending on which way you are going.

The from-to based style of the trip-based approach can furthermore be used by traffic experts to analyze the frequencies of turning left, right, or continuing straight ahead at an intersection.

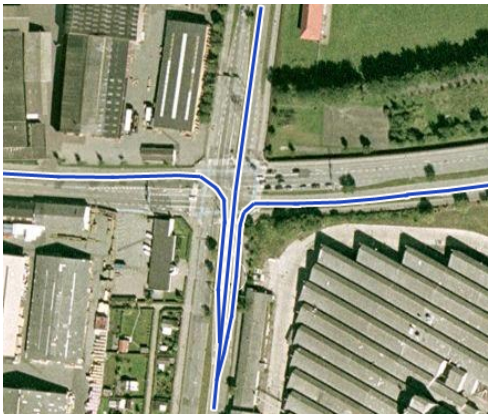

Figure 6: 4-way intersection.

6.1 Map Matching

In order to be able to determine to which road segment we attach observations, we use two different types of map matching. The first and most simple approach is a simple nearest neighbour query as used in the point-based approach.

With trip-based data, we can use the context of previous and following observations to increase map-matching accuracy. While we have designed and implemented our own approach to this, it is inspired by [3].

The map matching works on a single trip at a time. For each observation, the map-matching algorithm finds all candiate road segments within a certain distance. This distance can be varied, but the accuracy of the GPS observations should be reflected herein. When given these candidate segments, the map-matching algorithm works by dividing the entire trip into smaller sub-problems whenever the algorithm encounters an observation with only a single candidate segment. If an observation only has a single road segment within the specified distance, the algorithm assumes that there is certainty that the observation is located on that road segment. Based on this assumption, the algorithm can use these observations as fixpoints. Every observation between two fixpoints will have two or more candidates, or none at all. If they have none, it is a special case, which we will address later. Otherwise, in order to be able to exclude some of the candidates, the algorithm runs a shortest-path algorithm between each set of fixpoints. Given the list of road segments returned by the shortest-path algorithm, the map-matching algorithm removes all candidates not in that list from each observation. If an observation still has more than one candidate (the observation is near an intersection), the closest road segment is chosen.

This algorithm works best with data that is recorded at a high frequency; otherwise, the probability of the shortest path being the actual route decreases and data would have to be discarded.

In Algorithm 2, the map-matching algorithm for trip-based data is listed. As mentioned above, it takes a list of observations as input, and outputs the same list, with a segment id attached to each observation.

On line 2, a new instance of the *Redolist* class is declared. A *redolist* is a simple object that holds two observations, *first* and *last* as well as a list of observations, *nodes*. It has two object methods, *clear* and *solve*. The algorithm for the *solve* method will be discussed shortly. The *clear* method simply clears the *nodes* list and sets *first* and *last* to null. The *Save* method saves the particular observation.

The loop in line 4 iterates through each observation. Lines 5-6 add the observation to the *redolist* if the number of candidate road segments is greater than one (i.e. there is more than one road segment within a certain distance). If there is only a single candidate, on lines 8-12 we can now solve the *redolist*, which is done in Algorithm 3. On lines 14-15, if there are no candidates for an observation, we call *TripMatch* recursively with the rest of the trip, effectively splitting the trip in two. Here, *ArraySlice* is a auxiliary function that takes an array and returns the last part of it, based on an integer value. Finally, we break the loop as the rest of the trip will be processed by the recursive call.

Algorithm 2: Recursive function TripMatch

```
   input  : A list of observations, Trip
   output : A list of map-matched observations
 1 begin
 2     redolist = new Redolist ;
 3     count = 0;
 4     foreach obs in Trip do
 5         if sizeof(obs.candidates) > 1 then
 6             redolist.add(obs);
 7         else if sizeof(obs.candidates) = 1 then
 8             redolist.last = obs;
 9             redolist.solve();
10             redolist.clear();
11             redolist.first = obs;
12             obs.save();
13         else
               // If sizeof(obs.candidates) < 1
14             TripMatch (ArraySlice (Trip, count+1));
15             break;
               // Rest of trip is solved in recursive call
16 end
```

In Algorithm 3, we solve the *redolist* created in Algorithm 2. The *first* and *last* variables are our fixpoints, as discussed earlier. On line 2, we find the shortest path between those two points. The loop starting on line 3 iterates through the nodes of the *redolist* (the observations with more than one candidate segment). On line 4, we remove all candidates not on the shortest path between *first* and *last*. Next, if we still have more than one candidate, we chose the candidate that is closest to the line. If we have removed all candidates, on line 7-8, we skip saving the observation, as it cannot be used due to the fact that it does not have a valid candidate road segment. Finally, on line 9 we save the observation with the new candidate.

We have chosen the above algorithm, as we believe it has the best performance to accuracy ratio of the algorithms, we considered. It does have one disadvantage; it requires two fixpoints, where we are certain about their locations on the network. However, before the first fixpoints on the trip, and after the last, there will be observations not between two fixpoints. Therefore, the above algorithm can only discard those observations.

An alternative approach would be to modify the above

Algorithm 3: Function Redolist.Solve

```
   input  : A Redolist object, list
   output : Null
 1 begin
 2     path = ShortestPath(list.first, list.last);
 3     foreach obs in list.nodes do
 4         RemoveMissing(obs.candidates, path);
 5         if sizeof(obs.candidates) > 1 then
 6             obs.candidates = GetClosest(obs);
 7         else if sizeof(obs.candidates) < 1 then
 8             continue;
               // Ignore observation as no credible candidates can
                  be found
 9         obs.save();
10 end
```

algorithm in order to enable the use of all available data. Instead of requiring two fixpoints, we could create a virtual fixpoint based on the first observation on the trip. By running the shortest-path algorithm on every candidate segment of this observation, we would have several possible routes between the first fixpoint and the first observation on the trip. The set intersection of these routes would yield a shortest path with a high likelihood of being correct. The path variable on line 2 of Algoritm 3 is set to this value. If the first observation does not have a candidate on the path, it is removed. This also applies to any consecutive points until an observation with a candidate on the path is found. Once this observation is reached, the algorithm continues as normal. We have opted not to implement these modifications in our project for a number of reasons: Primarily, the number of observations discarded for each trip is very small (less than 1% on average). Secondly, execution time is increased as the shortest-path algorithm, which is expensive time wise, must be run several times over as compared to just once in Algorithm 3.

6.2 Storage

trip_fact	
PK	**id**
	trip_number
FK1	date_id
FK2	vehicle_id
FK3	driver_id
FK4	source_id
FK5	time_id
FK6	segment_from
FK7	segment_to
FK8	road_condition_id
	permitted_speed
	update_frequency
	travel_time

Figure 7: Fact table for storing trips.

To store trips we have introduced a separate fact table, *trip_fact*, which can be seen in Figure 7. The *trip_fact* table contains the same foreign keys as *gps_fact* introduced in Section 4, and three additional fields, *segment_from*, *segment_to* and *travel_time*. The fields *segment_from* and *segment_to* replace *segment_id* from *gps_fact*, and store a the fraction of the trip that covers *segment_from*. The field *travel_time* stores the time, in seconds, taken to get from the beginning of *segment_from* to the beginning of *segment_to*. The field *trip_number* corresponds to the same field in the *gps_fact* table. The field attributes observations and entries in *trip_fact* with an identifier so that trips can be extracted in their full length after they have been stored. The field could also be a foreign key to an empty trip dimension.

trip_travel_time	
PK	**id**
FK1	**segment_from**
FK2	**segment_to**
	travel_time
	heading
	<interval>
	count

Figure 8: Table for storing trip-based travel times.

Figure 8 shows the table used to store travel times for the trip-based approach. In this approach, several travel times must be stored, one for each neighbouring road segment, compared to just two in the point-based approach. However, this solution is still more space efficient than the point-based approach, as we only store one row per segment per trip, whereas the point storage model stores several observations for each of those segments. In other words, the point-based approach needs all the observations while the trip-based approach only saves the information about how long it took to get from one intersection to the other.

Theoretically, the space we save can be estimated as follows:

The average segment length in our network is 185 meters, and the average speed for all observations is 29.3 $\frac{km}{h}$. This means that the average number of observations per segment per trip is $\frac{0.185km}{29.3\frac{km}{h}} * 3.6\frac{s}{h} \simeq 22.7$. This means that, in our network, there will theoretically be 22.7 rows in the *gps_fact* table for each row in the *trip_fact* table. In fact, a quick count reveals that on 6667 trips we have 1933187 rows in *gps_fact* versus 123189 rows in *trip_fact*, which corresponds to a ratio of 15.7:1. The difference between 22.7:1 and 15.7:1 can be explained by the fact that on shorter road segments there will be fewer observations, which means that we save less space. In the city where our trips are located, the road segments are usually shorter.

6.3 Splitting Trips

To fit the segment-based style of both the point-based approach and the storage model, the trips found while map matching must be split into smaller pieces. This introduces a problem. While the sampling rate of the trip-based approach is no lower than 10 seconds, there is no guarantee that observations are available precisely at the beginning and end of each road segment, which is needed in order to measure the travel time precisely.

The situation is shown in Figure 9. To find the actual travel time between intersections, interpolation is used and the resulting travel time is then stored in the *trip_fact* table. In the figure, the time at the intersection (the big circle) can be found by interpolating between observation 2 and 3. By finding distance A and B, the percentage of the distance located on the road segment labelled "from" can be found by dividing distance A by distance $A + B$. That percentage can then be multiplied by the difference in time between the two points to find the time at the intersection. Algorithm 4 calculates a *per segment* travel time based on the trips found earlier using interpolation.

Algorithm 4 works by finding the last observation on a road segment and the first observation on the next road segment and interpolating between those observations as previously described. On lines 2 and 3 the algorithm is initialized and on line 4, it loops over all the observations. The first observation on the road segment, *first*, is set on line 8, this will

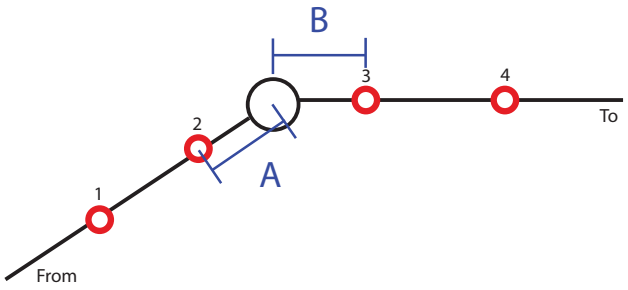

Figure 9: Interpolating times at intersections.

Algorithm 4: Trip split

```
   input  : A list of observations, O
 1 begin
 2     curr = start = O[0];
 3     end = O[sizeof(O)-1];
 4     for i=1; i<count(O); i++ do
 5         prev = O[i-1];
 6         curr = O[i];
 7         if !isset(first) then
 8             first = curr;
 9         if prev.segment_id != curr.segment_id then
10             last = prev;
11             curr_interpol = interpolate(last, curr);
12             if last.segment_id != start.segment_id then
13                 store(first.segment_id, curr.segment_id,
                       curr_interpol - prev_interpol);
14             prev_interpol = curr_interpol;
15             first = curr;
16             unset(last);
17 end
```

be one of the observations used in the interpolation. When a change in *segment_id* is detected on line 9, *last*, is found by backtracking one observation. Using the last observation on one road segment and the first observation on the next road segment, a time can be interpolated on line 11.

If the segment of observation *last*, which is the first of the two observations used in the interpolation, is not the first road segment in the trip, stored in *start*, then a travel time is stored on line 13. This is of course because the part of the trip from the first observation to the first intersection cannot be used since the travel time from the first observation back to the previous intersection cannot be extrapolated without introducing uncertainty.

On lines 14-16 the variables are changed for the next iteration.

The trip algorithm will work without using the above interpolation algorithm, and given that our data is recorded at one observation per second, it will not have a large impact on the estimate accuracy. However, with data recorded at a lower rate, the interpolation algorithm should improve accuracy. We will test this claim in Section 8. If there is an identical traffic flow on both segments involved in the estimation, the interpolation technique is accurate. However, this is hardly ever the case. There are special cases, where the interpolation will actually decrease accuracy. This occurs when the traffic situation is such, that the last of the two segments has a slower traffic flow than the first, and the two observations used to determine the interpolated time are recorded such that the first observation is very close to the transition to the next segment, and the other is far from it.

6.4 Calculating Travel Times

After interpolation, the next step is to calculate travel times from the trips stored in the *trip_fact* table. Algorithm 5 calculates travel times in almost the same manner as Algorithm 1 does for the point-based approach. The difference lies in that the trip-based approach uses the from-to approach to finding travel times. For each road segment, travel times to all its neighbours must be calculated. On line 4 of Algorithm 5 the neighbours for the current road segment, r, are found. The following SQL statement finds the neighbours:

```
SELECT unique(pol_id)
FROM connections
WHERE conn_id IN (
    SELECT conn_id
    FROM connections
    WHERE pol_id=r)
  AND pol_id!=r;
```

When the neighbours are found, travel times to each neighbour can be found on lines 5-10 in the same way as in the point-based algorithm. Here the *getAvg* function is expressed by the following SQL statement:

```
SELECT avg(travel_time)
FROM trip_fact
WHERE segment_from=r
  AND segment_to=n
  AND <interval>;
```

The length of the road segment and the travel time is calculated as in the point-based algorithm. On line 10 the calculated travel time is stored in the database as described in Section 6.2.

Algorithm 5: Trip-Based Algorithm

```
   input  : A list of road segments, R
   input  : A list of intervals, I
 1 begin
 2     avg = 0, len = 0, tt = 0;
 3     foreach r in R do
 4         neighbours = getNeighbours(r);
 5         foreach n in neighbours do
 6             foreach <interval> in I do
 7                 avg = getAvg(r, n, <interval>);
 8                 len = getLength(r);
 9                 tt = (len/avg)*3.6;
10                 store(tt, r, n, <interval>);
11 end
```

7. VALIDATION

In this section, we will validate the trip-based and point-based approaches. Validation of the two approaches will be based on data from the *Spar på farten* data set. We have found two separate routes, one route in the center of the city of Aalborg, where we know that traffic jams are frequent and one route on one of the major roads not in the central city. The two routes are shown in Figure 10. The red route we will refer to as "Vesterbro" and the blue route as "Sohngårdsholmsvej".

As we do not have the means to perform an extensive data gathering operation to find the actual travel times on the routes, for example by setting up a license plate recognition system and recording traffic data, we will verify our estimates by using data from our data set. Based on the *Spar på farten* data set we have manually determined the travel time for the two routes at two different periods of time by analyzing the data and making sure that any outliers are not used. For the Vesterbro route we are using *Monday-Thursday 6:00-8:00* and *Monday-Thursday 12:00-14:00* to maximize the volume of data while maintaining a stable travel time within the time period. For the

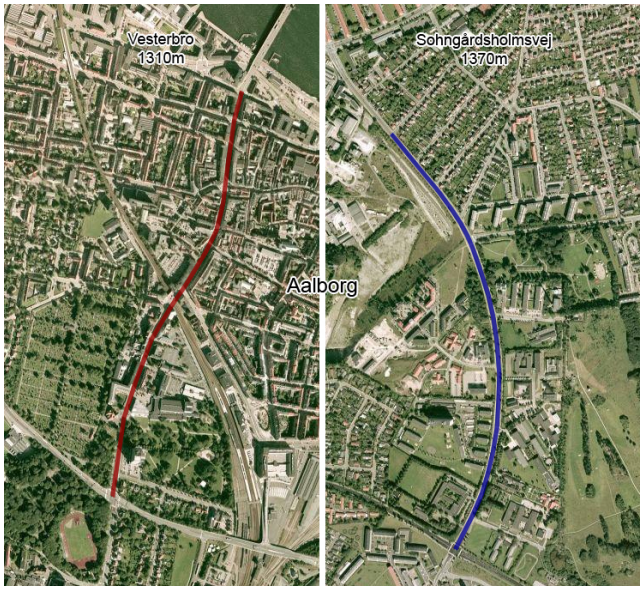

Figure 10: The routes used in validation.

Sohngårdsholmsvej route the intervals are *Monday-Thursday 6:00-8:00* and *Monday-Thursday 14:00-16:00*. These travel times are compared to the travel times calculated for the same routes and time periods by the point-based and trip-based approaches. Both the trip-based approach and the point-based approach will be based on the data from the *Spar på farten* data set, using the one-second sampling rate. The results of the validation can be seen in Tables 1 and 2. The entries labeled "Actual" are the travel times manually determined. The entries marked with "Same" are only based on the data used to determine the actual travel times. The entries marked with "All" are based on the entire *Spar på farten* data set which should give an idea of how the two approaches should perform in practice. All times are in seconds.

Method	Direction	Time 6-8	Time 12-14
Actual	North	150.2	210.7
Trip (Same)	North	151.2	210.7
Trip (All)	North	160.1	213.9
Point (Same)	North	140.4	213.4
Point (All)	North	158.3	253.4
Actual	South	205.0	229.5
Trip (Same)	South	205.0	229.6
Trip (All)	South	210.7	249.3
Point (Same)	South	209.0	222.4
Point (All)	South	175.0	249.8

Table 1: Vesterbro

As expected the trip-based approach estimates the travel times nearly 100% correct using the same data set while the point-based approach is slightly off. It is important also to note that the trip-based approach performs better than the point-based approach using the entire data set.

Theoretically, the trip-based approach should not deviate from the actual travel time using the same data foundation. In Tables 1 and 2, we see that the trip-based approach deviates very little from the actual travel time. The actual travel times are determined by analyzing how long it took to get from the beginning of the route to the end, which is the same approach as the trip-based algorithm uses, only it does so for each segment instead of the entire trip. The small deviations are due to the interpolation of observations, which

Method	Direction	Time 6-8	Time 14-16
Actual	North	105.3	140.8
Trip (Same)	North	105.3	140.8
Trip (All)	North	115.5	141.7
Point (Same)	North	117.5	140.7
Point (All)	North	134.7	146.3
Actual	South	90.8	84.7
Trip (Same)	South	92.1	85.0
Trip (All)	South	95.0	102.0
Point (Same)	South	94.0	85.1
Point (All)	South	107.6	134.8

Table 2: Sohngårdsholmsvej

may be slightly off, as we will show in 9. The point-based approach treats all points as equal which means that for the point-based approach to be 100% correct all the points must be evenly distributed along the route, which explains the small deviation between the actual travel time and the travel time estimated by the point-based approach. This is because the average speed will be based on an unevenly distributed set of observations but when converting speed to travel time an even distribution is assumed.

Using the entire data set the point-based approach deviates even more. This might have several reasons. First of all it might be that the data in the entire data set is more unevenly distributed. It might also be, that the small error accumulates as more data is used. Secondly, it might be data from single, unconnected observations; these would be discarded in the trip-based approach. Finally, in the trip-based approach, outlier detection is aided by the fact that individual observations are groups together in trips. This makes it easier to detect stops not related to traffic conditions. In the point-based approach observations are considered independent which makes it much harder to detect outliers, yielding an inferior data foundation for the point-based approach. In our validation we manually removed three trips from "Vesterbro 6-8, South", where the cars had clearly been parked on a nearby parking lot for an extensive period of time. This process could be automated by an efficient outlier detection algorithm. Such algorithms are already well documented, and thus outside the scope of this article. The effect of doing so clearly shows in the results, where the difference between the actual travel time and the travel time calculated by the point-based approach is 14.6%.

The numbers in Tables 1 and 2 seem to indicate that the trip-based approach has an average error of 0.3% (from 0% to 1.4%), while error of the point-based approach is 3.4% (from 0.07% to 10.4%) using only the same data set.

8. EXPERIMENTS

In this section, we will perform a number of experiments in order to be able to compare and evaluate the point-based and trip-based algorithms. First, we will investigate how fast the two algorithms become accurate if gradually fed more data. We will also investigate the effect of lowering the sampling rate for the *Spar på farten* data set. The native sampling rate of the data set is one observation every second. In our experiment, we run a series of test on a selected route, gradually lowering the sampling rate to one observation every two minutes. During the same experiment, we will investigate the effects of interpolation between observations, as mentioned in Section 6.3. Finally, we will compare the road network coverage of both approaches.

8.1 Data Quantity

In our first experiment, we will examine exactly how much

data is needed to provide an accurate estimate. For this purpose, we will run a series of tests, each time increasing the amount of data on which the estimate is based. We will use the same data as in our verification, more specifically the data from Vesterbro between 6 and 8, in the northern direction. We chose this data due to the fact that both algorithms provide roughly the same estimate. This is important when we want to compare how fast the algorithms converge on the final estimate. It may not be the same as the actual average travel time for our selected trips, but this is due to the fact that the selected trips in this case are not entirely representative of the entire data set.

The initial run of the algoritms will rely only on data from the first week of the year. Each consecutive run will add another week until we have the full 52 weeks of the year.

In Figure 8.1, we show the results of the experiment. The horizontal axis is the number of weeks included in the estimate, while the vertical axis is the estimated travel time in seconds. Notice that the horizontal axis starts at week 26. This is due to the fact that we do not have data for the entire route for the first 25 weeks of the year. From week 9 to week 26 only a part of the route is covered by the data in the chosen time period. The solid line is the trip algorithm, the dashed line is the point-based algorithm.

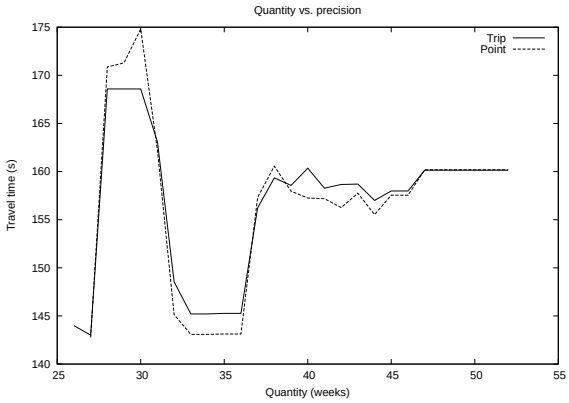

Figure 11: Quantity of data: Effect on accuracy

From the graph, it can be seen that both algorithm converge on 160 seconds, which is the result from the validation. It can also be seen that the accuracies of the algorithms improve drastically until week 47, where both lines remain stable. From the experiment, we can see that the trip algorithm is generally more accurate than the point algorithm, with the exception of weeks 37 and 38. It can also be seen that both algorithms follow the same tendency, which is due to the fact that the data foundation is the same. It is interesting to note that the trip algorithm performs better than the point algorithm, even with a relatively small amount of data.

In the experiment, we added a single week at a time. As such, it is not entirely linear, which may skew the graph. In Figure 12, we show the actual number of observations used to create Figure . As the figure shows, the last observation is in week 47. Until that point, the line is approximately linear, the largest exception being around week 33, where 500 observations are added at once.

Figure 13 shows the same experiment including the *Bektra* data set. Instead of Vesterbro between 6:00 and 8:00, we use Vesterbro for the entire day to get enough data in the *Bektra* data set. The figure shows that the trip-based and point-

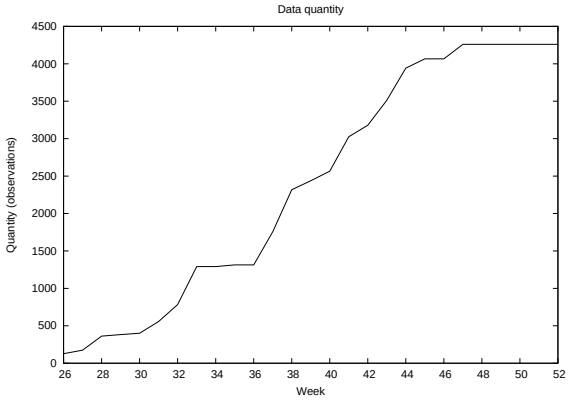

Figure 12: Data Quantity

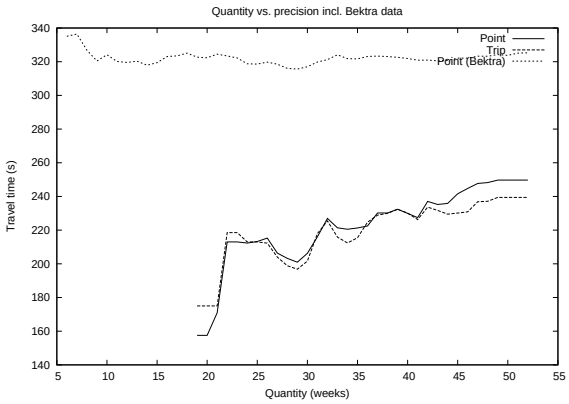

Figure 13: Accuracy of the Bektra data set

based approaches still agree on the travel time. The *Bektra* data set, however, is far from the two others; the travel time is much higher and the shape is much more stable. The travel time is so high that we suspect that the data is biased because it is collected by taxis. We know that the taxis have dedicated stopping locations along Vesterbro, which seems to have a large impact on the travel time. The shape of the *Bektra* line in the figure lacks the same general form as the *Spar på farten* data set. The *Spar på farten* data set exhibits the tendency that we would expect, so a more thorough analysis of the *Bektra* data set is needed, but that is outside the scope of this article.

8.2 Sampling Rate

For our second experiment, we investigate the effects of lowering the sampling rate for our trip-based approach. We vary the sampling rate from one observation per second, which is the sampling rate of the Spar på farten data set, up to and including one observation every other minute as in the Bektra data set. Again, we have chosen the same trips as in the validation: Versterbro, northern direction between 6:00 and 8:00 in the morning. This experiment does not include an estimate for the point-based approach, as it is not dependant on sampling rate.

The average length of road segments on Vesterbro is ap-

proximately 82 meters. Based on the average segment length, we can calculate a rough estimate of the sampling rate needed to get at least one observation per segment. Since the speed limit is 50 $\frac{km}{h}$, an average segment will take 5.9 seconds to traverse at the permitted speed. However, the average speed of all our observations on Vesterbro is 20,4 $\frac{km}{h}$. At this speed, it will take 14.5 seconds. From this rough estimate, we expect that sampling rates should be higher than once every 14-15 seconds in order to have at least one observation per segment on average. In order to provide an accurate estimate, it should be even higher, as many segments are shorter than the average. In the following, we will investigate the above speculations.

Figure 14 shows the result of the experiment. The horizontal axis is the sampling rate and the vertical axis is either travel time in seconds or a percentage. There are three completely horizontal lines. From top to bottom they are: Travel-time estimated by the naive approach, the actual travel-time, and a line to indicate 100 percent.

The grey dashed lines at the top of the figure are travel-times estimated by the trip-based algorithm. The top most line is without interpolation. Finally, the dotted grey line at the bottom of the figure indicates how many percent of the segments used in the estimation that had to revert to the naive approach due to lack of data on the segment.

As can be seen in the figure, the trip algorithm performs well with high sampling rates, but as soon as there is more than a few seconds between the samples, the accuracy is lowered substantially. As the naive percentage line indicates, the trip algorithm quickly becomes unable to predict travel times for many of the segments as the sampling rate is lowered. In experiments, we lowered the sampling rate to one observatation every 120 seconds, but after 30 seconds, nearly all estimates uses the naive fallback method, and as such, they are not interesting. Therefore, the figure only ranges from 1 to 30.

The naive algorithm is actually better if the sampling rate is lower than 6 observations per minute (10 seconds between observations). At this point, we are forced to revert to the naive algorithm for more than half of the segments. At 30 seconds between each observation, nearly all segments are calculated without using the actual data. This is not surprising considering that with a trip that takes 150 seconds, and thereby consists of 150 observations, removing all but every 30th of those leaves just 5.

During our experiments, we verified our intuition that lowering the sampling rate effects short segments first. This is due to the fact that the probability of an observation being recorded at short segments is lower than for long segments, caused by the cars spending more time on the long segments. In our experiment, short segments are weighted the same as long segments for the naive percentage. This means that the percentage seems higher than if we had used the total road length for comparision instead of segment count. While short segments acccount for many of the naive fallbacks, they do not have a large impact on total travel time, as short segments per definition only make up a small part of the total travel time, given an even distribution of traffic.

The reason that the trip-based approach becomes more inaccurate than the naive approach lies in the problem of determining when the trip algorithm is accurate. When we know the actual travel time, it is very clear that the trip algorithm is less accurate, but naturally, when estimating the time, we do not know the actual travel time. The trip algorithm is unable to exclude observations that are slower than the actual time, as it is impossible to determine wheter the long travel time is due to inaccuracies of the algorithm or if it is caused by slow traffic.

From the experiment, it can be seen that the trip-based approach requires a high sampling rate. This is a clear disadvantage of the trip-based approach, as any inaccuracy due to the low sampling rate is an irreversible error, as adding more data with the same low sampling rate will not improve the trip-based estimate. The point-based approach does not perform well with small amounts of data, but here, adding more data will rectify the situation, as it is simply based on an average.

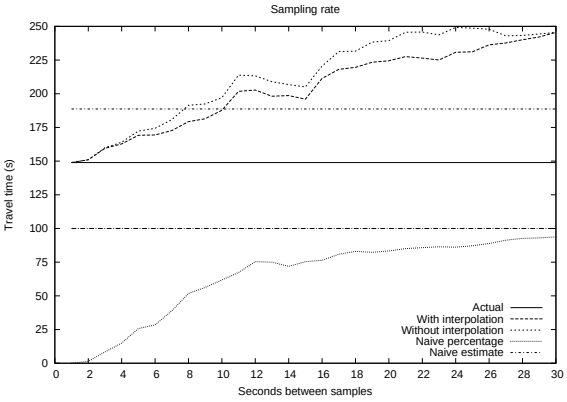

Figure 14: Sampling rate: Effect on accuracy

8.3 Interpolation

In the previous experiment, aside from lowering the sampling rate, we also investigated the effect of disabling interpolation in our trip-based algorithm. The results are also shown in Figure 14, and clearly shows that interpolation increases accuracy. As we expected, the improvement is greatest as the sampling rate is lowered. At 30 seconds between samples, both approaches use the naive fallback method, and interpolation is no longer used for the vast majority of segments. Because of this, the increase in accuracy is decreased to zero.

8.4 Data Coverage

The aim of our final experiment is to investigate the data coverage of the point and trip algorithms. We simply count the segments which are covered by enough data to provide a reasonable estimate. For the trip-based approach, this means that the algorithm must have been able to interpolate a travel-time fromgq a trip with a sufficient sampling rate. For the point-based approach, we define coverage as having atleast 10 observations on a segment, as segments with only a few observations cannot be accurately estimtated.

In total, there are 1773 segments with at least a single observation in our system. Of those, 1520 have at least 10 observations for the point-based approach. The trip-based approach covers 1481 of those segments. As the above figures show, the difference in coverage between the two approaches is roughly 3%.

9. DISCUSSION

In this section, we will discuss some of the design choices we have made. They are mostly concerned with the algorithms of the trip-based aprroach, as it is by far the most complex algorithm.

9.1 Segmentation of Trips

One dilemma we faced during the project is the concept of dividing the trips into segment-based travel times versus

using entire trips.

One argument is that the accuracy when using entire trips would be greater, as an estimate would be based on the exact same stretch of road as the trips on which it is based, which means that the timings of the intersections are taken into consideration. As previously mentioned, we have chosen to split the trips into smaller pieces. This is done for a number of reasons: First, we find it nearly impossible to rely on trips with the exact same path as the estimate target for long journeys. The probability of finding a trip with the exact same segments traversed has an inverse proportional relationship with the number of segments. As such, any reasonable algorithm would divide the trips into smaller segments, perhaps between cities or major roads. As an example, take Figure 15. The motorway and bridge between Odense and Kolding is an major arterie in Danish domestic traffic. As such, it is often traversed and should therefore be well covered in any data set covering the region. In order to illustrate a point, let us then assume that someone would like to know the travel-time estimate for the route between Kolding and Stige via Odense, the roads between Stige and Odense being much less used than the motorway. Unless we divide the trips into smaller parts, we would only use all trips traveling exactly the same route, meaning that all trips stopping in Odense or continuing south or east would not be considered. The trips that were used in this estimation would most likely be very accurate, but in order to have a nuanced picture of reality, many cars would have to drive that exact trip many times in order to be able to provide an estimate for all hours of day, weekends etc.

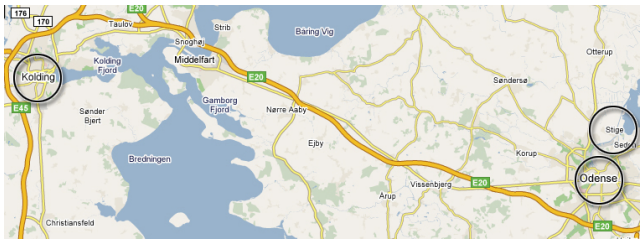

Figure 15: An example trip.

If the trip was split between Kolding and Odense, and Odense and Stige, however, all trips on both stretches of road could be used individually. By our logic, there is no reason to stop there, however. By dividing each trip into segments, we get the best of both worlds. We retain the ability to use only the trips, that traverse the exact same segments as the target path if we want to, yet we can use all available data as we calculate the travel time per segment. As we have shown in our validation, our trip-based algorithm will return the alomst exactly same travel time as one that measure the start and end time of a trip.

Another way of looking at it is if you have 100 trips from Kolding to Odense, and 20 trips from Odense to Stige, the approach that splits the trips into smaller pieces would use the union of the two sets of trips, whereas insisting on using entire trips would result in using only the intersection.

9.2 Lowering Sampling Rate

In our experiments, we would like to have compared the performance of the two algorithms when lowering the sampling rate gradually from once per second to once every two minutes. But in the context of the point-based algorithm, sampling rate is irrelevant in any respect other than increasing it would provide more observations. As such, it is not the frequency of the observations, but rather the quantity.

Another related issue that stems from lowering the sampling rate is the fact that any error made in the estimation of a segment is not summed up as one would think, but rather they cancel each other out. This is best illustrated in Figure 16.

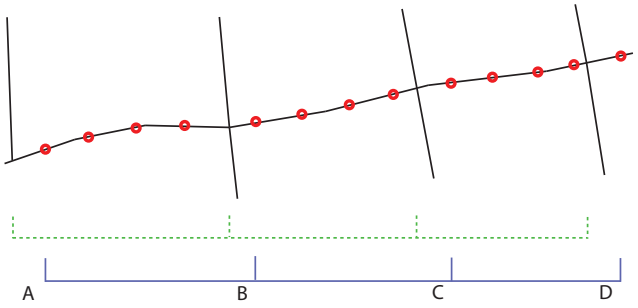

Figure 16: Sources of inaccuracy of the trip algorithm.

Here, the dots are observations along a road network illustrated by the black lines. If we wanted to determine the travel time for each individual segment, we would use the time stamps of the first observation for each segment if we choose not to use interpolation. The solid lines at the bottom of the figure indicate these times. The dashed lines indicate the actual times when the car passes from one segment to another. Had we chosen to use interpolation, the principle would be the same, but the solid lines would be in a different location. For the sake of simplicity, we will ignore interpolation in this example.

In the figure, both the solid and dashed line form a continuous time period. There are no gaps or overlaps in the travel times for either line. Take the first set of lines at A and B. The actual travel-time interval starts earlier than the estimated interval, but it also finishes later. Because there is no gap leading to the next set of intervals between B and C, any error made near B is cancelled out. The same applies to C, when we continue. In other words, the total error made is not A+B+C+D, as one would expect, but rather, it is simply the difference between the error made near A and D. A more mathematical way of describing this would be as follows:

We use the following methodology to generically describe a point in time. A_A is the time on the dotted line in Figure 16, where the first vertical line intersects the horizontal. In other words, when the observed car actually enters the first segment. E_B is the time on the solid line where the second vertical line intersects - or the time of the first recording on the second segment. A_x is the actual travel time, E_x is the estimate. S_n is segment number n.

The error per segment is:

$$S_1 : (A_B - A_A) - (E_B - E_A)$$

$$S_2 : (A_C - A_B) - (E_C - E_B)$$

$$S_3 : (A_D - A_C) - (E_D - E_C)$$

Total:

$$\begin{aligned} S_1 + S_2 + S_3 =& (A_B - A_A) - (E_B - E_A) + (A_C - A_B) - \\ & (E_C - E_B) + (A_D - A_C) - (E_D - E_C) \\ =& A_B - A_A - E_B + E_A + A_C - A_B - \\ & E_C + E_B + A_D - A_C - E_D + E_C \\ =& A_B - A_B + E_B - E_B + A_C - A_C + \\ & E_C - E_C + A_D - A_A + E_A - E_D \\ =& A_D - A_A + E_A - E_D \end{aligned}$$

Rewritten, as the error is an absolute percentage:

$$|(A_D - A_A) - (E_D - E_A)|$$

The above result means that the only possible deviance from the actual travel time is at either end of the trip. In

addition, the differences will always counteract each other. What this means in an actual environment is that while each individual segment might be inaccurate, when estimating the travel time for several segments, there are only two places where an error might be introduced.

9.3 Fallback Strategy

In the previous sections, we introduced two approaches for travel-time estimation on single road segments. Both of these methods require a certain amount of data to work, but in a road network, data shortage can easily occur on smaller or less used roads. Both the point-based and trip-based approach require the same amount of data to converge, but the trip-based approach also requires a high sampling rate. For this reason, the point-based approach is more flexible in that it can be applied in more cases than the trip-based approach. On the other hand, the trip-based approach is faster, requires less storage and is more accurate. The trip-based approach is also able to distinguish between the different destinations from a given road segment.

Combined, the two approaches complement each other so that a travel-time estimate can always be calculated. The trip-based approach is used when possible, but if the sampling rate is not sufficient for the trip-based approach, the point-based approach can be used as a fallback method.

For the solution to be feasible in practice, more fallback methods are needed. In our previous article [13], we introduced the naive approach, that was able to provide an estimate for segments, where no data was available. This method suffers from a greater inaccuracy than the data-based approaches, just as it is not able predict rush hours.

However, the naive approach is a necessity, if we want to be able to predict travel times even when little or no data is available. For example on little used road segments or if no data is available initially.

The solution is to use a naive approach, which defines the travel time for a road segment as the length of the road divided by the allowed speed, multiplied by a constant factor. Either this constant factor can be determined manually, or it can be based on an average retrieved from the database. It is used as it is unrealistic to assume that cars can drive at the permitted speed, as there will inevitably be intersections, lane merges or other circumstances that will reduce the average speed. Either this factor can be completely constant, that is to say, determined statically by someone with domain knowledge, or it can be found by comparing the average speed to the permitted speed on segments where data is available. Ideally, the map used should have an indication of to which zone each segment belongs. This is based on the assumption that city zones opposed to the country generally have more speed reducing obstacles in the form of aforementioned intersections, lane merges, etc.

Other possible fallback methods include neighbourhood approaches, where the travel time for a road segment lacking GPS data is based on the travel time for neighbouring roads of the same road type. This has the disadvantage that although the roads are close and of the same type, it is still impossible to know whether the road actually exhibits the same traffic patterns. Similar to this, one might argue that if a given segment had enough data in one direction, it would be possible use this data in both directions.

One flaw remains in both above ideas: if there is not enough data available on a particular segment, yet there is a segment nearby with data available, there must be a reason for this. The difference in data quantities could very likely be caused by a difference in the amount of traffic. If traffic is not the same, then the average speed is unlikely to be the same either. The same principle applies to segments, where data is only available in one direction of traffic.

9.4 Calculating Route Travel Times

In practice, travel times are seldom based on single road segments but rather on entire roads or a start and destination pair. Finding the travel time for an entire road or a route is as simple as summing up the road segment travel times involved. For the point-based approach this can be done by a simple SQL statement using a list of road segments, <route>, and <interval> as earlier:

```
SELECT sum(travel_time)
FROM point_travel_time
WHERE segment_id IN (<route>)
  AND <interval>;
```

For the trip-based approach the from-to based style requires that for each road segment in the route, its successor is defined in order to find the corresponding travel time. Algorithm 6 sums up the travel time for a route from the beginning of the first road segment to the beginning of the last road segment.

Algorithm 6: Trip-Summation Algorithm

```
   input  : A list of road segments, R
   input  : An interval, <interval>
   output : The total travel time, total
1  begin
2      total = 0;
3      for r = 0; r ¡ sizeof(R) - 1; r++ do
4          from = R[r];
5          to = R[r+1];
6          tt = getTT(from, to, <interval>);
7          total += tt;
8  end
```

In Algorithm 6 the function *getTT* is expressed by the SQL statement:

```
SELECT avg(travel_time)
FROM gps_fact
WHERE segment_from=from
  AND segment_to=to
  AND <interval>;
```

Finding the actual route between two points can be done by using a shortest-path algorithm. We are using a variant of the A^* algorithm adapted for our road-network structure. Using a timed version of the A^* algorithm the shortest path between two points can be found by using the travel times stored in the *point_travel_time* and *trip_travel_time* tables thus finding the fastest path instead of the shortest path. This has not yet been implemented in our code, but it is a matter of replacing the body of a single function, which now only returns the naive estimate.

9.5 Trips or Points

While the trip-based approach has many advantages over the point-based approach, it is not without problems. As such, there will be circumstances that make the point-based approach better than the trip based approach. First and foremost of these circumstances is the sampling rate. As we showed in Section 8.2, lowering the sampling rate has a great impact on the accuracy of the trip-based algorithm. Therefore, we cannot recommend using the trip-based algorithm, if there is more than 3 or 4 seconds between observations. As we discussed earlier, any error made by the trip-based algorithm due to the low sampling rate is irreversible, whereas the point-based approach improves as more data is added, regardless of sampling rate.

In Section 8.1, we showed that the trip-based approach was more accurate than the point-based approach with the same amount of data. This is surprising, as we would assume

that the average value used to estimate travel times in the point-based approach would require little data to provide a good estimate. While this is true to some extent, if the data is sampled once per second, the trip-based approach is better. The point-based approach is better as soon as the sampling rate drops. This leads to the discussion of when to use the point-based approach instead of the trip-based. We have already established that a high sampling rate will provide the best estimate, but it also comes at a cost. In several scenarios, it is imaginable that the data costs money to transfer. For example, some GPS receiver has rely on either UMTS or GPRS technology to transfer the data from the receiver to a central server, as is the case with *Bektra*. As such solutions come at a high bandwidth cost, it might be more feasible to get less data from more cars in order to be able to cover an entire road network instead of having the same amount of observations centered around the most often used roads of fewer drivers. In the ideal case, drivers would upload the data using a hard line, which is both typically faster and cheaper. Depending on the scenario, both methods can therefore be the best option.

10. CONCLUSION AND FUTURE WORK

In this article, we have developed two approaches to travel-time estimation, the point-based approach and the trip-based approach, both based on our earlier work in [13]. Both approaches estimate travel times for a road network using GPS data provided by different sources. To store GPS data, the road network, and the information that allows us to filter out specific GPS data, we have designed a data warehouse flexible enough to accommodate future additions to our travel-time system including vehicle and driver information, road conditions and addresses. Both of our approaches have been implemented to the extent that we need in order to verify them. Using a number of experiments, we have verified that both the point-based and trip-based approaches work as intended. Furthermore, we have performed a number of experiments to show how the two approaches perform. As it turns out, the trip-based approach converges at the same rate as the point-based approach, but oscillates less, which means that it is more precise than the point-based approach, even when less data is available. We have discussed how the point-based and trip-based approaches can be combined in practice to calculate travel times and have provided one possible strategy for doing so.

The main contribution of this article, the trip-based approach, is more space efficient, more precise and faster than the point-based approach and certainly an improvement over using a naive travel time. As a side benefit, the trip-based approach can be used to examine the frequencies of turning right, left or continuing in an intersection. This is used by traffic experts when analyzing intersections.

The core of our travel-time system is now in a state where we can begin to estimate travel times, but as always, much more can be done. The following is a short list of the furture work we would like to look into.

Storing Travel Times and Intervals. A topic we have only briefly touched is the storage of travel times after they have been calculated. This topic is closely related to the intervals used in Section 5 and 6 as the table structure of tables *point_travel_time* and *trip_travel_time* depends on the time intervals chosen. For example, one might choose to store a travel time for each road segment for each hour of the entire year. No matter what granularity is chosen it might be possible to group periods of similar travel times together to save space. For example, if the travel time for Mondays 8-9 and Mondays 9-10 are similar enough, they can be combined to a single travel time, Mondays 8-10. This topic is described in more detail in [13].

Using Auxiliary Information. Travel times depend on a number of factors that we have mentioned earlier but not used in our running system. Information about drivers, vehicle types, weather and road conditions may improve the current travel-time estimate. As mentioned earlier, drivers and vehicles drive by specific patterns, roads are periodically under construction and periods of extreme weather (for the season) occur, these cannot be predicted, but they can be used to filter out any odd data at a later point.

Dynamic Map. One of the problems in dealing with road networks is the fact that roads change slowly over time. In order to consider this we first need a mechanism to determine when old data is no longer useful, as the road has changed. Furthermore, the map needs to be maintained, either manually or - preferably - automatically.

Outlier Detection. As briefly mentioned in Section 7, outlier detection is needed to remove trips and observations that are not representaive for the entire data set. Such trips include those where the car has been stopped at the side of the road for an extensive period of time or trips where the driver for some reason drives much slower than the rest of the traffic. There are two options; either the observations are discarded at load time or simply not used when estimating travel times.

ACKNOWLEDGEMENTS

We would like to thank Agne Brilingaite and Christian S. Jensen for providing a map matching algorithm on which we can base our system. Also, we thank *Bektra* [1] and the *Spar på farten* [6] project for providing invaluable GPS data.

REFERENCES

[1] BeKTra. http://www.behovstyrettrafik.dk, June 2007.

[2] Sotiris Brakatsoulas, Dieter Pfoser, and Nectaria Tryfona. Modeling, storing, and mining moving object databases. In *IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium (IDEAS'04)*, pages 68–77, 2004.

[3] Agne Brilingaite, Christian S. Jensen, and Nora Zokaite. Enabling routes as context in mobile services. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems (GIS'04)*, pages 127–136, 2004.

[4] Francois Dion and Hesham Rakha. Estimating spatial travel times using automatic vehicle identification data. *Intelligent Transportation Systems*, pages 1–30, 2003.

[5] ESRI. http://www.esri.com, June 2007.

[6] Spar Paa Farten. http://www.sparpaafarten.dk, June 2007.

[7] Barbara Frith, David Pearce, and Tom Sutch. The highways agency journey time database. *Road Transport Information and Control*, pages 98–105, 2004.

[8] GPSY.com. http://www.gpsy.com/gpsinfo/gps-faq.txt, June 2007.

[9] Ralf Hartmut Güting, Victor Teixeira de Almeida, and Zhiming Ding. Modeling and querying moving objects in networks. *The VLDB Journal*, pages 165–190, 2006.

[10] C. Hage, C.S. Jensen, T.B. Pedersen, L. Speicys, and I. Timko. Integrated data management for mobile services in the real world. In *VLDB 2003: Proceedings of 29th International Conference on Very Large Data Bases*, pages 1019–1030, 2003.

[11] Christian Overgaard Hansen. Traengselsindikator for biltrafik. *Trafikdage*, pages 1–10, 2004.

[12] Anthony Harrington and Vinny Cahill. Route profiling: putting context to work. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing (SAC'04)*, pages 1567–1573, 2004.

[13] Anders Forum Jensen and Troels Villy Larsen. Travel-time estimation in road networks using gps data, 2006.

[14] Christian S. Jensen, Anders Friis-Christensen, Torben B. Pedersen, Dieter Pfoser, Simonas Šaltenis, and Nectaria Tryfona. Location-based services: A database perspective. In *ScanGIS*, pages 59–68, 2001.

[15] Christian S. Jensen, Augustas Kligys, Torben Bach Pedersen, and Igor Timko. Multidimensional data modeling for location-based services. *The VLDB Journal*, pages 1–21, 2004.

[16] C.S. Jensen, T.B. Pedersen, L. Speicys, and I. Timko. Data modeling for mobile services in the real world. In *Advances in Spatial and Temporal Databases*, pages 1–9, 2003.

[17] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. Finding fastest paths on a road network with speed patterns. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, pages 1–10, 2006.

[18] Wei-Shinn Ku, Roger Zimmermann, Haojun Wang, and Chi-Ngai Wan. Adaptive nearest neighbor queries in travel time networks. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems (GIS'05)*, pages 210–219, 2005.

[19] Yanying Li and Mike McDonald. Link travel time estimation using single gps equipped probe vehicle. *Intelligent Transportation Systems*, pages 932–937, 2002.

[20] Otto Anker Nielsen. Analyse af traengsel og hastigheder vha. gps-data. *Trafikdage*, pages 1–21, 2003.

[21] Dieter Pfoser, Nectaria Tryfona, and Agnes Voisard. Dynamic travel time maps - enabling efficient navigation. In *SSDBM '06: Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, pages 369–378, 2006.

[22] Cesar A. Quiroga and Darcy Bullock. Travel time studies with global positioning and geographic information systems: an integrated methodology. *Transportation research. Part C : Emerging technologies*, pages 101–127, 1998.

[23] J. Rice and E. van Zwet. A simple and effective method for predicting travel times on freeways. *Intelligent Transportation Systems*, pages 227–232, 2001.

[24] Michael A.P. Taylor, Jeremy E. Woolley, and Rocco Zito. Integration of the global positioning system and geographical information systems for traffic congestion studies. *Transportation research. Part C : Emerging technologies*, pages 257–285, 2000.

[25] Igor Timko and Torben Bach Pedersen. Capturing complex multidimensional data in location-based data warehouses. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems (GIS'04)*, pages 147–156, 2004.

[26] S. Turner, R. Margiotta, and T. Lomax. Lessons learned: Monitoring highway congestion and reliability using archived traffic detector data. *Mobility Monitoring Program - Year 4*, pages 1–36, 2004.

[27] B.S. Yoo, S.P. Kang, and C.H. Park. Travel time estimation using mobile data. In *Proceedings of the Eastern Asia Society for Transportation Studies, Vol. 5*, pages 1533–1547, 2005.