Dynamic Travel Time Maps - Enabling Efficient Navigation

Dieter Pfoser
RA Comp. Tech. Institute
Akteou 11
GR-11851 Athens, Greece
pfoser@cti.gr

Nectaria Tryfona
Talent SA
Karytsi Square 4A
GR-10561 Athens, Greece
tryfona@talent.gr

Agnès Voisard
Fraunhofer ISST
Mollstr. 1
D-10178 Berlin, Germany
Agnes.Voisard@isst.fraunhofer.de

Abstract

Routing plays an ever-important role in a society that relies heavily on individual means of transportation. Although efficient algorithmic solutions for navigation exist, an accurate and reliable weight database that forms the basis of an acceptable algorithmic solution is missing. This work defines algorithms and data management techniques that allow the derivation of dynamic weights from collected Floating Car Data (FCD). Weights reflect the speed associated with a piece of road at a certain time. A collection of such historical data is used to capture trends in travel time behavior according to temporal variations. Based on large amounts of travel time data, edge-based weights in the form of time-varying characteristic travel times are derived. Since the available tracking data does not cover the entire road network, several methods are defined to compensate for the lack of data and to guarantee complete coverage. A dynamic weight database, the Dynamic Travel Time Map (DTTM) is defined and implemented as a spatio-temporal data warehouse to manage the characteristic travel times and to compute dynamic weights efficiently. An experimental evaluation establishes not only the efficiency of the proposed approach but also shows its applicability in a realistic context, using actual GPS vehicle tracking data for the road network of Athens, Greece.

1 Introduction

The major limitation in conventional routing and navigation systems is the unreliable travel time database associated with the underlying road network. With such a network, only static weights as derived from road category and speed limits are used to calculate the fastest or shortest path for a given trip.

In this work, we propose the *Dynamic Travel Time Map* (DTTM) as a means to efficiently supply dynamic weights that are derived from a collection of relevant historic travel times by exploiting the causality between historic and current traffic conditions.

We use *Floating Car Data* (FCD) and specifically the tracking data component as a means to derive travel times for a road network. FCD is a by-product in fleet management applications and given a minimum number and uniform distribution of vehicles, this data can be used for accurate traffic assessment and also prediction. Map-matching the tracking data produces travel time data related to a specific edge in the road network.

The DTTM that we present is realized by means of a spatiotemporal data warehouse. The basic fact table contains the collected travel time data. Using a temporal and a spatial dimension, travel times are aggregated to provide appropriate dynamic weights for the entire road network. Redundancy is the key element to achieve a good coverage. The more tracking data is available (the larger the number of tracked vehicles at any given time), the better is the overall coverage and accuracy of the DTTM.

As far as routing data and static weights are concerned, such data is now available for most places on Earth from a few vendors (e.g., NAVTEQ© and TeleAtlas©). In such databases, even though a traversal estimate (link-based speed types) is associated with all road portions (based on average speed, itself based on speed limit for such road categories), such data are static in the sense that they do not take real traffic conditions into account. In a first attempt to use FCD in this context, [11] illustrates the use of floating car data to assess traffic conditions.

The paper is organized as follows. Section 2 gives background information on routing algorithms and, together with their limitations, points out the importance of a dynamic weight database. Section 3 introduces the concept of characteristic travel time as derived from historic travel time collections. Section 4 defines a spatiotemporal data warehouse that implements the dynamic travel time map. Section 5 evaluates the performance of the DTTM to compute dynamic weights in terms of accuracy, cost and speed of computation. Finally, Section 6 gives conclusions and directions for future research.

2 Background and Motivation

A major accuracy problem in conventional navigation systems is due to the unreliable travel time associated with the underlying road network. A network is made of basic pieces usually referred to as links. A link is an atomic road portion such as a piece of road between two intersections or a ramp onto an highway. With a link is associated a value that expresses its level in a hierarchy of roads (also known as arterial level). The link notion is introduced to reflect the connectivity between roads of possibly different importance. With a link is then associated a speed, which is, per default, static. In the following, we outline the basic principles behind routing algorithms before advocating the use of *dynamic* routing algorithms.

2.1 Algorithmic Solutions to Navigation

A road network, is modeled as a directed graph $G = (V, E)$, whose vertices V represent intersections between links and edges E represent links. Additionally, a real-valued weight function $w : E \rightarrow \mathbf{R}$ is given, mapping edges to weights. In the routing context, such weights typically correspond to speed types derived from road categories or based on average speed measurements. However, what is important is that such weights are static, i.e., once defined they are rarely changed. Besides, such changes are rather costly as the size of the underlying database is in the order of dozens of Gigabytes.

A shortest path from vertex u to v is defined as any path p with weight $w(p) = \delta(u, v)$, i.e., a weight that is minimal [2]. The formal description of the problem as well as shortest path algorithms are available in the literature.

Assuming that all edge weights in the graph are non negative, $w(u, v) \geq 0$, the basic problem of finding the shortest path between two vertices u and v (single-pair shortest path problem) can always be solved by applying a uniform cost algorithm (also known as greedy or Dijkstra's algorithm [4]).

Although Dijkstra's algorithm is complete and optimal in that it finds the shortest path to any vertex u from a starting vertex s, it is an uninformed search algorithm that can be improved by exploiting knowledge on the way to reach the goal. In that context, informed search algorithms ("best-first search") were proposed, among them a pure heuristic-based algorithms and the A* algorithm [6], which combines the uniform cost algorithm and a heuristic-based algorithm. The A* algorithm differs from Dijkstra's algorithm in that for the selection of $u \in V - S$, it uses the minimum shortest-path estimate, i.e., the cost of the path so far, $d(u)$, in combination with the *estimated cost to the goal*, $h(u)$. For the latter, a heuristic is needed that estimates this cost. For routing problems such as the shortest-path, the straight-line distance, or, Euclidean distance to the goal, is a good heuristic function (in the sense that it does not overestimate the cost to reach the goal) and can also easily computed. The estimated cost of the cheapest solution through u, $f(u)$, is as follows.

$$f(u) = d(u) + h(u) \tag{1}$$

With admissible[1] h, A* is optimal [3], however, it runs out of memory quickly as it keeps all generated nodes. Memory-bounded algorithms were introduced, which guarantee optimal solutions even though some solutions are eliminated. Among them we can cite Iterative Deepening A*, MA*, Simplified Memory-Bounded A* (SMA), IE, or Tabu Search (see [9] for a comprehensive description of all these algorithms). Their optimality relies on the size of a maintained heap and on the pruning strategy. Navigation systems usually use bi-directional, memory-bounded algorithms based on A*.

2.2 Dynamically-assigned Weights

Although the various algorithmic solutions cited above are still subject to further research, the basic place for improving solutions to routing problems is the underlying weight-based database $DB(w(u, v))$. Currently, data from vendors correspond to static weights, or, *link-based speed types*, which are derived from the respective road category (or arterial levels) and its associated speed limit, or a speed type determined by costly road-side surveys.

Our work aims at making such a weight database fully dynamic with respect to not only a spatial but also a temporal argument. The idea is to derive dynamic weights from historic traffic assessment based on sensor measurements in the form of *floating car data (FCD)*. Using the causality between historic and current traffic conditions, weights - defined based on temporal parameters - will replace static weights such as the ones currently in use and will induce impedance in traversing some links.

2.3 Dynamic Weights and Routing

In a typical collection scenario, FCD is available in close to real time [11]. In connection with a dynamic weight database, such a live traffic assessment has serious implications for the implementation of any type of routing algorithm not only (i) by increasing the accuracy of the result but also (ii) by potentially changing the algorithm itself and, thus, the way a routing solution is computed.

[1]A function f is admissible if it does not overestimate the cost to reach the goal.

A routing solution has an associated temporal validity, i.e., start time and estimated time to destination. Using dynamic weights based on historic travel times, this weight database at best is current at the time the routing solution is computed. Assuming a live FCD collection scenario, any discrepancy between newly recorded travel times and dynamic weights would require a recomputation of the routing solution.

An existing variant of the A* algorithm allows for changes to the world after the initial routing solution is computed, i.e., . The D* (Dynamic A*) algorithm [12] is intended for use when no complete information, i.e., weights, is initially available. As shown in a robot path planning context, the D* algorithm can be used with incomplete or *inaccurate* information [13]. Given an initially course "map", the robot detects its precise environment as it moves along. Translated to the navigation context this means that as a vehicle is presented with an initial routing solution based on a dynamic weight database, this database gets refined with now-relevant travel time data collected from vehicle fleets during the temporal validity of the routing solution. The more accurate the prediction of actual traffic conditions by the dynamic weight database, the better will the initial routing solution be and the fewer changes will be required during the execution phase.

Another domain in which dynamic weights can prove their usefulness is dynamic vehicle routing in the context of managing the distribution of vehicle fleets and goods. Traditionally, the only dynamic aspect were customer orders. Recent literature however mentions also the traffic condition and thus travel times as such an aspect [5].

3 Travel Time

Travel times are obtained by relating tracking data to edges in a road network. The approach to do so and what it takes to achieve a good travel time coverage for the entire road network are detailed in the following section.

3.1 Vehicle Tracking Data

Floating car data (FCD) refers to using data generated by one vehicle as a sample to assess to overall traffic conditions ("cork swimming in the river"). Typically these data comprise basic vehicle telemetry such as speed direction and most importantly the position of the vehicle in the form of *vehicle tracking data*. Having large amounts of vehicles collecting such data for a given spatial area such as a city (e.g., taxis, public transport, utility vehicles, private vehicles) will create an accurate picture of the traffic condition in time and space [11].

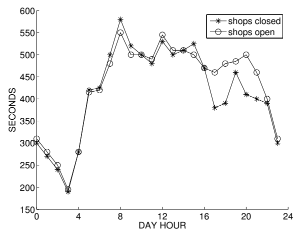
Matching the tracking data to a road network, we obtain travel times. Since FCD is typically obtained using GPS positioning, the associated measurement error in connection with the sampling rate require us apply specific map-matching algorithms to perform this task. The algorithm we used in the experimental evaluation of Section 5 is reported in [1].

The travel times are finally recorded as sets of time intervals associated with edges in the road network.

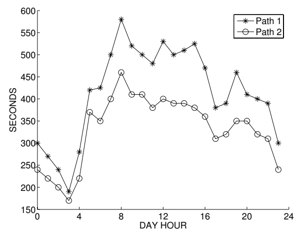
3.2 Travel Time Derivation

The collection of historical travel time data provides a strong basis for the derivation of dynamic weights provided that one can establish the causality of travel time with respect to time (time of the day) and space (portion of the road network).

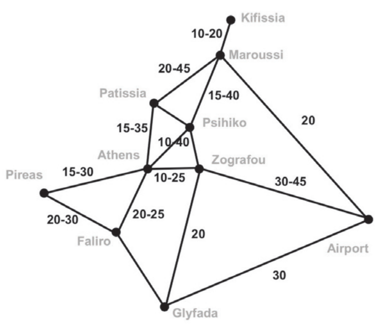
Temporal causality establishes that for a given path, although the travel time varies with time, it exhibits a recurrent behavior. An example of temporal causality is shown in Figure 1(a). For the same path, two travel time profiles, i.e., the travel time varying with the time of the day, are recorded for different weekdays. Although similar during nighttime and most daytime hours, the travel times differ significantly during the period of 16h to 22h. Here, on one of the days the shops surrounding this specific path were open from 17h to 21h in the afternoon.



(a) Temporal causality



(b) Spatial causality



(c) Travel time example

Figure 1. Relating travel times

Spatial causality establishes that travel times of different edges are similar over time. An example is given in Figure 1(b). Given two different paths, their travel time profiles are similar. Being in the same shopping area, their travel time profile is governed by the same traffic pattners, e.g., increased traffic frequency and thus increased travel times from 6h to 20h, with peaks at 8h and around noon. Figure 1(c) gives an overall example, illustrating travel time fluctuations in a road network.

Overall, discovering such temporal and spatial causality, affords hypothesis testing and data mining on historic data sets. The outcome is a set of rules that relate (cluster) travel times based on parts of the road network and the time in question. A valid hypothesis is needed that selects historic travel time values to compute meaningful weights.

In this work, for the purpose of defining and evaluating a tool that allows for the fast computation of dynamic weights, we adopt a simple hypothesis with respect to spatial causality that relates travel times (i) based on spatial proximity and (ii) road category. This is based on the observation that roads of the same category, e.g., neighborhood roads that are close to each other, exhibit similar travel time patterns. The specific road categories used in this work are described in Section 3.4.

3.3 Characteristic Travel Time

A problem observed in the example of Figure 1(b) is that travel times, even if causality is established, are not readily comparable. Instead of considering absolute travel times that relate to specific distances, the notion of relative travel time ρ is introduced, which for edge e is defined as follows,

$$\rho(e) = \frac{\tau(e)}{l(e)} \tag{2}$$

with $\tau(e)$ and $l(e)$ being the recorded travel time and edge length, respectively. [2]

Given a set of relative travel times $P(e)$ related to a specific network edge e and assuming that these times are piecewise independent, the *characteristic travel time* $\chi(P)$ is defined by the triplet cardinality, statistical mean, and variation as follows,

$$\begin{aligned} \chi(P) &= \{|P|, E[P], V[P]\} && (3) \\ E[P] &= \sum_{\rho \in P} \frac{\rho}{|P|} && (4) \\ V[P] &= \sum_{\rho \in P} \frac{(\rho - E[P])^2}{|P|} && (5) \end{aligned}$$

The critical aspect for the computation of $P(e)$ is the set of relative travel times selected for the edge e in question based on temporal and spatial inclusion criteria $I_T(e)$ and $I_S(e)$, respectively.

$$P(e) = \{\rho(e^*, t) : e^* \in I_S(e) \wedge t \in I_T(e)\} \tag{6}$$

[2]An alternative perception of ρ is that of *inverse speed.*

$I_S(e)$, a set of edges, contains typically the edge e itself but can be enlarged as we will see in the next section to include further edges as established by an existing spatial causality between the respective edges. $I_T(e)$, a set of time periods, is derived by existing temporal causality. Respective parameter choices are discussed in Section 3.5.

The characteristic travel time essentially represents a dynamic weight, since depending on a temporal inclusion criterion (e.g., time of the day, day of the week, month, etc.), its value varies.

3.4 Road Network

FCD and thus travel times are not uniformly distributed over the entire road network, e.g., taxis prefer certain routes through the city. To provide a dynamic weight database for the entire road network, a considerable amount of FCD is needed on a per edge basis, i.e., the more available data, the more reliable will be the characteristic travel time.

Availability of data is directly related to the road category. In this work, we adopt a simplistic model of major and minor roads, with *major roads* comprising highways and major urban streets and *minor roads* including smaller roads and alleys. Orthogonally, the amount of available FCD groups the edges into sets of *frequently traversed* and *non-frequently traversed* edges[3]. An empirical study established that for our specific data set (cf. Section 5.1), 99% of all frequently traversed edges belong to major roads. Figure 2 depicts the road network of Athens, Greece with the frequently traversed edges shown in black and dark gray.

3.5 Travel Time Computation Methods

While it is possible to compute the characteristic travel times for frequently traversed edges only based on data related to the edge in question, for the non-frequently traversed edges, with their typical lack of data, complementary methods are needed. The simplest approach is to substitute travel times by static link-based speed types as supplied by map vendors. However, following the causality principles as outlined in Section 3.2, the following three prototypical methods can be defined. The various approaches differ in terms of the chosen spatial inclusion criterion $I_S(e)$, with each method supplying its own approach.

Simple Method. Only the travel times collected for a specific edge are considered. $I_S(e) = \{e\}$.

Neighborhood Method. Exploiting spatial causality, i.e., the same traffic situations affecting an entire area, we use a simple neighborhood approach by considering travel times of edges that are (i) contained in an enlarged minimum bounding rectangle (MBR) around the edge in question and

[3]The threshold for categorizing edges as frequently traversed was set to 250 traversals.

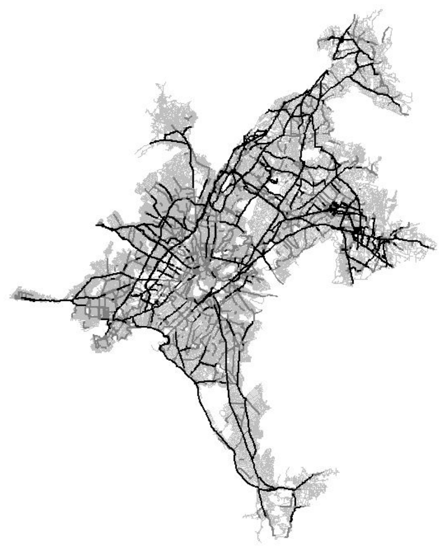
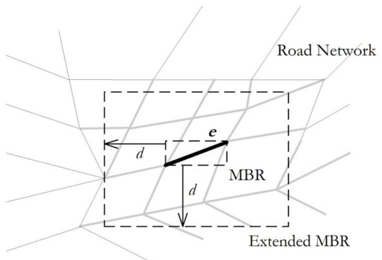
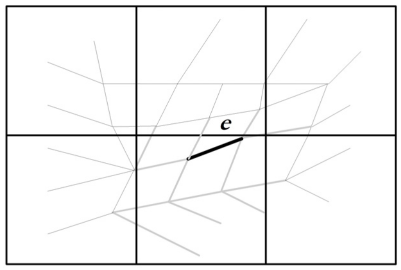


Figure 2. Road network and frequently traversed edges.



(a) Neighborhood Method



(b) Tiling Method

Figure 3. Characteristic travel time computation methods

(ii) belong to the same road category. Figure 3(a) shows a network edge (bold line) and an enclosing MBR (small dashed rectangle) that is enlarged by a distance d to cover the set of edges considered for travel time computation (thicker gray lines). $I_S(e) = \{e^* : e^* \text{ contained\_in } d\text{-expanded MBR}(e) \wedge L(e^*) = L(e)\}$, with $L(e)$ being a function that returns the road category of edge e.

Tiling Method. Generalizing the neighborhood method, a fixed tiling approach for the network is used to categorize edges into neighborhoods. It effectively subdivides the space occupied by the road network into equal sized tiles. All travel times of edges belonging to the same tile and road category as the edge in question are used for the computation of the characteristic travel time. Figure 3(b) shows a road network and a grid. For the edge in question (bold line) all edges belonging to the same tile (thicker gray lines) are used for the characteristic travel time computation. $I_S(e) = \{e^* : e^* \in Tile(e) \wedge L(e^*) = L(e)\}$

Both, the Neighborhood and the Tiling Method, are effective means to compensate for missing data when computing characteristic travel times. Increasing the d in the Neighborhood Method, increases the number of edges and thus the potential number of relative travel times considered. To achieve this with the Tiling Method, the tile sizes have to be increased. As we will see in the experimental evaluation (Section 5), both parameters have to be used with caution not to compute inaccurate characteristic travel times.

The Neighborhood method is expected to be computationally expensive since the computation of a characteristic travel time requires the retrieval of respective travel times for all the edges contained in an edge neighborhood (cf. Figure 3(a)) by using expensive ad-hoc spatial range queries. The Tiling Method as a simplified version of the Neighborhood method is expected to overcome this limitation by using pre-computed tiles and, thus, less expensive database queries. The question that remains to be answered is whether characteristic travel times and, thus, dynamic weights can be as efficiently computed as static weights. The following section proposes a data management approach based on data warehousing, which should have a competitive performance as will be established in Section 5.

4 Dynamic Travel Time Map

Manipulating large amounts of travel time data in relation to a road network and routing algorithms requires efficient data management techniques. For this purpose, the *Dynamic Travel Time Map* (DTTM), a spatiotemporal data warehouse for travel time data is introduced.

4.1 Requirements

The basic requirement to the DTTM is the efficient retrieval of characteristic travel times on a per-edge basis.

Based on the choice of the various inclusion criteria, Section 3.5 defines three computation methods. The DTTM needs to support each method in kind.

Sketching the query processing of the various computation methods should help in deriving the requirements to a DTTM. The Simple Method retrieves relative travel times relating to the edge in question. In case of the Neighborhood Method, the spatial inclusion criterion is interpreted as range query to identify the neighboring edges. For each such edge, relative travel times have to be retrieved. The Tiling Method uses as a spatial inclusion criterion a regular subdivision of space. For each edge that is within the same tile as the edge in question, relative travel times have to be retrieved to compute the respective characteristic travel time.

Our computation methods use arbitrary temporal and spatial inclusion criteria. This suggests the use of a data warehouse with relative travel times as a data warehouse fact and space and time as the respective dimensions. Further, since the tiling method proposes regular subdivisions of space, one has to account for a *potential lack of travel time data* in a tile by considering several subdivisions of varying sizes. The following section details the resulting data model.

4.2 Data Warehouse Design

The multidimensional data model of our data warehouse implementing the DTTM is based on a star schema. Figure 4 shows the schema comprising five fact tables and two data warehouse dimensions.

4.2.1 Dimensions

The two data warehouse dimensions relate to time, TIME_DIM, and to space, LOC_DIM, implementing the respective granularities as described in the following.

Spatial subdivisions of varying size can be seen as subdivisions of varying granularity that form a dimensional hierarchy. We adopt a simple spatial hierarchy with quadratic tiles of side length 200, 400, 800, and 1600 meters respectively, i.e., four tiles of xm side length are contained in the corresponding greater tile of $2x$m side length. Consequently, the spatial dimension is structured according to the hierarchy *edge*, *area_200*, *area_400*, *area_800*, *area_1600*, .

Should little travel time data be available at one level in the hierarchy, one can consider a higher level, e.g., area_400 instead of area_200. Section 5.3 will show how the accuracy of the characteristic travel time is affected by this aggregation.

Obtaining characteristics travel times means to relate individual relative travel times. In this work, the underlying temporal granularity of *one hour* is used, i.e., all relative travel times that were recorded for a specific edge during the same hour are assigned the same timestamp.

The temporal dimension is structured according to a simple hierarchy formed by the *hour of the day*, 1 to 24, with, e.g., 1 representing the time from 0am to 1am, the *day of the week*, 1 (Monday) to 7(Sunday), *week*, the calendar week, 1 to 52, *month*, 1 (January) to 12 (December), and *year*, 2000 - 2003, the years for which tracking data was available to us.

4.2.2 Facts

The measure that is stored in the fact tables is the characteristic travel time χ in terms of the triplet $\{|P|, E[P], V[P]\}$ (cf. Section 3.2).

The fact tables comprise a base fact table EDGE_TT and four derived fact tables, AREA_200_TT, AREA_400_TT, AREA_800_TT, and AREA_1600_TT, which are aggregations of EDGE_TT implementing the spatial dimension hierarchy. Essentially, the derived fact tables contain the characteristic travel time as computed by the Tiling Method for the various extents.

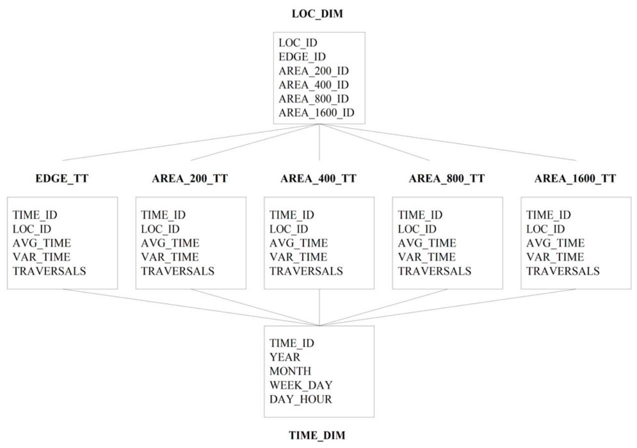


Figure 4. Data Warehouse Schema

4.3 Aggregation

While it is rather straightforward of how to compute characteristic travel times for the base fact table (EDGE_TT) based on relative travel times, aggregating characteristic travel times along the spatial and temporal dimensions needs some explanation.

In order to illustrate formally how aggregation can be performed, let P be a set of n relative travel times, $P = \{\rho_1, \rho_2, \ldots, \rho_n\}$, that are stored outside the data warehouse. Also, assume a set of k subsets of P, $S = \{S_1, S_2, \ldots, S_k\}$,

$S_i \subseteq P, 1 \leq i \leq k$, with the property $S_1 \cap S_2 \cap \ldots \cap S_k = \varnothing$. For each subset S_i we only store the characteristic travel time triplet $\{|S_i|, E[S_i], V[S_i]\}$.

In the data warehousing context, the formulation of the S_i subsets and their description through three aggregated measures constitutes a level of summarization. Now, assume that we summarize one level further. Let $I = \{1, 2, \ldots k\}$, and assume m disjoint subsets of I, $J_i \subseteq I$, $1 \leq i \leq m$, $J_1 \cap J_2 \cap \ldots \cap J_m = \varnothing$. Sets J_i help us to define the next level of summarization. Let $C = \{C_1, C_2, \ldots, C_m\}$, $C_i \subseteq S, 1 \leq i \leq m$, be a set of subsets of S such that $C_i = \bigcup_{j \in J_i} S_j$. It is clear that the property $C_1 \cap C_2 \cap \ldots \cap C_m = \varnothing$ holds. Set C expresses a higher level of summarization.

The characteristic travel time $\chi(C) = \{|C_i|, E[C_i], V[C_i]\}$ for a given level of summarization can be computed based on the respective characteristic travel times of a lower level, $\chi(S_j)$, without using the initial set of characteristic travel times P as follows.

$$|C_i| = \sum_{S_j \in C_i} |S_j| \tag{7}$$

$$E[C_i] = \frac{\sum_{S_j \in C_i} |S_j| \cdot E[S_j]}{|C_i|} \tag{8}$$

$$V[C_i] = \frac{\sum_{S_j \in C_i} |S_j| \left(V(S_j) + E[S_j]\right)}{|C_i|} - E^2[C_i] \tag{9}$$

While the derivation of cardinality $|C_i|$ and mean $E[C_i]$ follow from basic statistics, deriving the formula for variance $V[C_i]$ is more involved and shown in the following.

Using $V(Y) = E(Y^2) - E(Y)^2$, $V[C_i]$ can be written as

$$\begin{aligned} V[C_i] &= E[C_i^2] - E^2[C_i] & (10)\\ &= \frac{\sum_{S_j \in C_i} \sum_{x_l \in S_j} x_l^2}{\sum_{S_j \in C_i} |S_j|} - E[C_i]^2 & (11) \end{aligned}$$

To derive Equation 9 the x_l^2 term in Equation 11 needs to be substituted by known values.

$$\begin{aligned} V[S_j] &= E[S_j^2] - E^2[S_j] & (12)\\ &= \frac{\sum_{x_l \in S_j} x_l^2}{|S_j|} - E^2[S_j] & (13)\\ \sum_{x_l \in S_j} x_l^2 &= |S_j| \left(V[S_j] + E^2[S_j]\right) & (14) \end{aligned}$$

In the context of our specific data warehouse design, Equations 7, 8, and 9 can be used to populate the tables AREA_200_TT, AREA_400_TT, AREA_800_TT and AREA_1600_TT by aggregating over EDGE_TT, AREA_200_TT, AREA_400_TT and AREA_800_TT, respectively. This property is critical for our data warehouse design of a DTTM since we need to compute aggregated characteristic travel times along the various temporal and spatial dimensions.

5 Experimental Evaluation

The objective of this section is to establish, one, the accuracy of the three travel time computation methods, two, the respective computation costs, and, three, the computation speed of static vs. dynamic weights.

5.1 Data and Experimental Setup

The data used in the experiments comprises ca. 26000 trajectories that in turn consist of 11 million segments. The data was collected using GPS vehicle tracking through the years 2000 to 2003 in Athens, Greece. A positional sampling rate of 30 seconds was used. [4]

By using the Incremental map-matching technique as described in [1], this trajectory data was mapped onto a vector road map of Athens comprising 108,000 vertices and 150,000 edges.

To evaluate our approach experimentally, we implemented the DTTM using an Oracle 9i installation. The database block size was set to 6KB and the database cache to 192MB. We used [10, 7] as guides for data warehouse implementation and performance issues.

5.2 Edges and Paths

Three paths of varying length and composition (frequently vs. non-frequently traversed edges) were selected to conduct the experiments in this section. Table 1 gives their characteristics in terms of length and percentage of frequently-traversed edges.

	Length (km)	FT %
Path 1	2.0	50
Path 2	4.5	42
Path 3	2.2	13

Table 1. Paths used in experimental evaluation

Similar to aggregation, the characteristic travel time for a path δ, $\chi(\delta)$, is computed based on the characteristic travel times $\chi(l_i)$ of all edges l_i that comprise the path $\delta = \{l_1, \ldots, l_n\}$ as follows[5].

[4]The vehicle tracking data was supplied by Emphasis Telematics, a cooperating telematics company and fleet management service provider.

[5]Do note that cardinality has no meaning in the path context

$$\chi(\delta) = \{\varnothing, E[\delta], V[\delta]\} \quad (15)$$

$$E[\delta] = \sum_{l_i \in \delta} E[l_i] \quad (16)$$

$$V[\delta] = \frac{1}{E[\delta]} \sum_{l_i \in \delta} V[l_i] \cdot E[l_i] \quad (17)$$

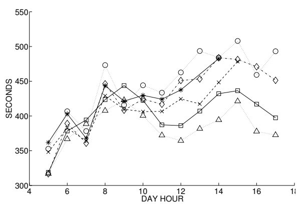
The parameter setting with respect to inclusion criteria is as follows. The temporal inclusion criterion I_T is the given hour of the day and considering all weekdays, e.g., all travel times collected between 8am and 9am from Monday to Friday. Relating to the DTTM of Section 4, the chosen temporal granularity of the data is the hour of the day, for days 1 to 5, of all months and years. The chosen spatial inclusion criterion I_S depends on the method that is used to compute the characteristic travel time. For frequently traversed edges, the Simple Method is used and only travel times relating to the edge itself are considered. For non-frequently traversed edges either the Tiling Method with respective tile sizes, or the Neighborhood Method is used. In the latter case, the expansion measure $d = 100m$ creates a "neighborhood" around the edge that approximately corresponds to the smallest tile size of the Tiling Method.

5.3 Accuracy

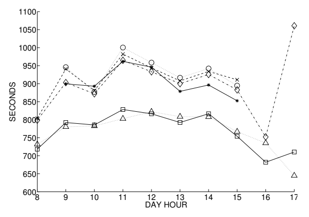
The accuracy of the three travel time computation methods is evaluated using the three example paths. Figure 5 and Figure 6 show the resulting travel time mean values and standard deviation (instead of variation).

With the Simple Method expected to produce the most accurate result, Figure 6 confirms this by showing the lowest standard deviation for this method. The Tiling Method produces the result with the second highest accuracy with a tile size of $200m \times 200m$ (*area_200*). Interestingly, the neighborhood method has the lowest accuracy (highest standard deviation) of all methods. The travel time mean value plots of Figure 5 reveal that the Simple Method and the Tiling Method - AREA200 produce similar values with slightly different accuracy. With the Tiling Method being able to compensate for missing data and *increase the availability of travel time information*, Figure 5 shows that with the Simple Method, travel times are available until 2pm, 3pm and 5pm, whereas by using the Tiling Method, they are available until 5pm, 5pm and 7pm for the three paths, respectively. The path with the smallest percentage of frequently-traversed edges (Path 3 in Figure 5(c)) has the biggest advantage in terms of data availability when using the Tiling Method.

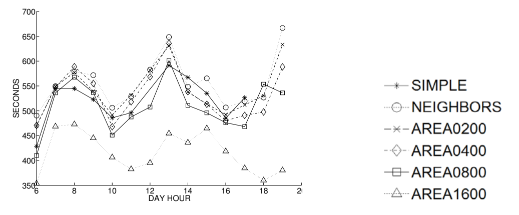
Regardless of the computation method used, both, the values for mean and standard deviation exhibit similar trends. A "smoothing effect" is obvious for the Tiling Method when using large tiles (*area_800* and *area_1600*) due to the large amount of travel time data available in these cases. Accordingly, the standard deviation increases.



(a) Path 1

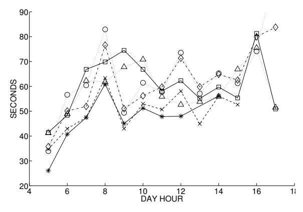


(b) Path 2

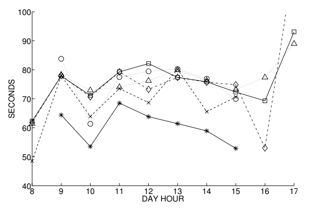


(c) Path 3

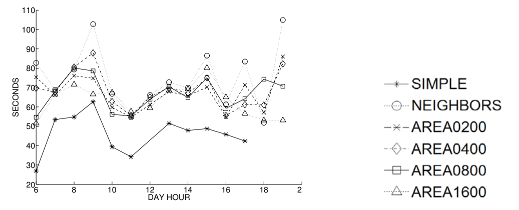
Figure 5. Characteristic travel times - Mean



(a) Path 1



(b) Path 2



(c) Path 3

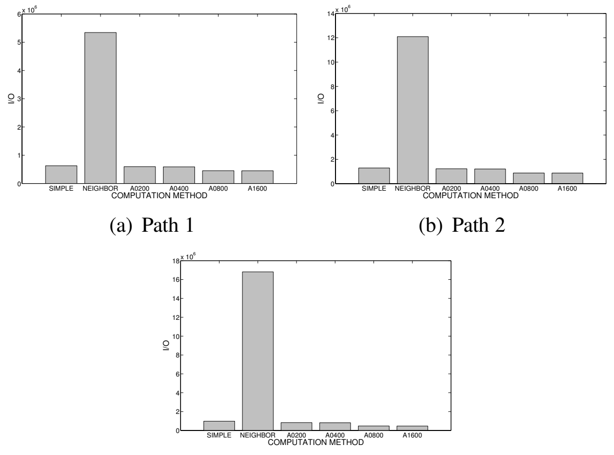
Figure 6. Characteristic travel times - Standard deviation

5.4 Computation Cost

It is expected that the travel time computation cost varies heavily with the respective method. To conduct the experiments, the following database queries are used. References to tables refer to the DTTM schema of Section 4.2.

Simple method. One query involving the base fact table is sufficient (EDGE_TT).

Neighborhood method. A spatial range query is used to identify neighboring edges. This query is issued to a



(a) Path 1 (b) Path 2

(c) Path 3

Figure 7. Computation cost - I/O operations

simple table that is not described in the schema of Section 4.2 containing only the geometries of the edges and the EDGE_ID. For each such edge, one query to the base fact table (EDGE_TT) is performed.

Tiling Method. The first query is used to identify the tile the edge belongs to. This relationship is kept in a simple table implementing the relationship AREA_ID, EDGE_ID. Subsequently, for each tile the corresponding fact table (AREA_[200|400|800|1600]_TT) is queried.

Figure 7 shows for each example path the computation cost in terms of number of I/O operations. We also measured CPU time, but omitted the charts since they showed the same relative results. As expected, the Neighborhood Method is the most expensive method. Although supported by a spatial index, a spatial range query to discover neighboring edges has to be executed for each edge in the path.

The performance of the Simple Method and the Tiling method are comparable. The Tiling Method performs best with large tiles. For a path with a small *number of frequently-traversed edges*, the Tiling method outperforms the Simple Method. The Simple Method needs only one query to the data warehouse, while the Tiling Method performs an additional query to determine the tile the edge belongs to. However, the dominant factor that greatly affects performance is the query to the fact tables. The Tiling Method uses fact tables that are aggregations of the base fact table. Consequently they contain fewer rows (cf. Table 2) and this results in an improved performance. Given a small number of frequently-traversed edges in the path, this advantage becomes more evident.

5.5 Dynamic vs. Static Speed Types

Section 5.3 established that several computation methods produce accurate travel times. Considering also their

Fact Table	Rows
EDGE_TT	5465682
AREA_200_TT	1045765
AREA_400_TT	631212
AREA_800_TT	344196
AREA_1600_TT	171401

Table 2. Fact table sizes

cost, the Tiling Method using a tile size of $400m \times 400m$ (*area_400*) turned out to be the best compromise.

To evaluate the feasibility of computing dynamic weights, the following experiment compares the computation speed of dynamic to static weights. The experiment utilizes a simple schema consisting of only one relation to compute the static weights. The table STATIC_TT(EDGE_ID, TT) contains random-generated static weights covering the entire road network (150k entries). Creating an index on EDGE_ID allows for an efficient retrieval. In contrast, dynamic weights are retrieved by means of the DTTM using the Tiling Method and granularity *area_400*. The query result comprises the characteristic travel time for this edge. Using a query load of 1000 queries, edges are randomly selected to compute the static and dynamic weights. A temporal inclusion criterion was chosen to only consider weekdays and to select the time of the day randomly.

	CPU (secs)	I/O
STATIC	0.002	4.5
DYNAMIC	0.018	32.7

Table 3. Performance measures for speed type computation

The results in Table 3 show that static weights can be computed *nine* times faster than dynamic weights. Still, in absolute terms, roughly 50 dynamic weights can be computed per second. Moreover, if we consider that for routing purposes edges are processed in succession, the Tiling Method allows us in most cases to reuse already computed values. This makes the costs shown in Table 3 a worst-case measure. In a practical setting, they will be much lower.

6 Summary and Research Directions

The availability of an accurate weight database is of crucial importance in the context of routing and navigation applications. This work proposes and develops a new data management technique, the *Dynamic Travel Time Map* (DTTM), to derive dynamic - in terms of time and space - weights based on collections of large amounts of vehicle tracking data. The DTTM is essentially a spatio-temporal data warehouse that allows for an arbitrary aggregation of travel times based on spatial and temporal criteria to efficiently compute characteristic travel times for a road network. Such on-the-fly computed travel times correspond to dynamic weights. Given that historic travel time information is not uniformly available for the entire road network, we devise two methods, namely the Neighborhood Method and the Tiling Method, which compensate for this lack of data such that the DTTM can provide dynamic travel times for the entire network. An experimental evaluation establishes the accuracy of travel times computed by the various methods as well as the cost of computation in terms of number of I/O operations. Finally, the Tiling Method, which gives the best compromise between accuracy and computation cost, is compared to an approach for computing static weights in terms of computations per second.

Our ongoing and future work are as follows. The dynamic speed types need to be delivered to the users requiring routing and navigation solutions. The DTTM we have proposed serves as underlying structure for any routing algorithm, such as a double-sided memory bounded A^* as it is in use now. We are currently investigating an adaptation of a variant of the A^* method that uses dynamic weighting [8] in conjunction with a weighted sum of $g + h$. Further, as the D^* algorithm has only been used in a robot path planning context, we are adapting this algorithm to be used for adaptive vehicle routing. The strategy for fetching and caching data plays a crucial role there. The distinction between short-distance and long-distance routing then needs to be taken into account. In this context, the distinction between on-line and off-line information delivery also needs to be distinguished and addressed with different techniques. An essential aspect for providing accurate dynamic weights is the appropriate selection of historic travel times. To provide and evaluate such hypothesis, extensive data analysis is needed possibly in connection with field studies.

Acknowledgments

This research is supported in part by the IXNILATHS project and the DROMEAS project (HRON-P), both funded by the Greek General Secretariat of Research and Technology (GSRT) and the TRANSIT project funded by the German Ministry of Education and Research (BMBF).

References

[1] S. Brakatsoulas, D. Pfoser, R. Sallas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *Proc. 31st VLDB conf.*, pages 853–864, 2005.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, second edition, 2001.

[3] R. Dechter and J. Pearl. Generalized Best-First Search Strategies and the Optimality of A*. *Journal of the ACM*, 32(3):505–536, 1985.

[4] E. W. Dikjstra. A Note on two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] B. Fleischmann, E. Sandvoß, and S. Gnutzmann. Dynamic vehicle routing based on on-line traffic information. *Transportation Science*, 38(4):420–433, 2004.

[6] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Science and Cybernetics*, 4(2):100–107, 1968.

[7] R. Kimball. *The Data Warehouse Toolkit*. John Wiley, second edition, 2002.

[8] I. Pohl. The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving. In *Proc. 3rd IJCAI conf.*, pages 12–17, 1973.

[9] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2003.

[10] B. Scalzo. *Oracle DBA Guide to Data Warehousing and Star Schemas*. Prentice Hall, 1st edition, 2003.

[11] R.-P. Schaefer, K.-U. Thiessenhusen, and P. Wagner. A Traffic Information System by Means of Real-time Floating-car Data. In *Proc. ITS World Congress*, Chicago USA, 2002.

[12] A. Stentz. Optimal and Efficient Path Planning for Partially-Known Environments. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 3310–3317, 1994.

[13] A. Stentz. Map-Based Strategies for Robot Navigation in Unknown Environments. In *Proc. AAAI Spring Symp. on Planning with Incomplete Information for Robot Problems*, pages 110–116, 1996.