METHOD AND DEVICE FOR DETERMINING A NAVIGATION PROFILE FOR A VEHICLE IN A GEOGRAPHICAL AREA

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to methods and devices for determining a navigation profile for a vehicle in a geographical area.

BACKGROUND

[0002] The quality of an e-hailing service which enables customers to hail taxis using their smartphones largely depends on how accurate the determination of navigation profiles, i.e. the routes taken or to be taken by the e-hailing service's vehicles is since the determination of these navigation profiles is the basis for various parameters which are necessary for efficient operation like the allocation of vehicles to trips and navigation of vehicles. However, road networks may be complex and there may be many parameters which have an impact on whether a road is taken or should be taken for a trip, e.g. whether a lot of traffic has to be expected on the road, there is a speed limit, the road is private, the road requires payment of a toll etc. Therefore, approaches for determining navigation profiles which allow accurate prediction of routes which are taken or which should be taken are desirable.

SUMMARY

[0003] Various embodiments concern a method for determining a navigation profile for a vehicle in a geographical area, including obtaining a road network for the geographical area, the road network including road segments connecting locations in the geographical area, initializing a navigation map by assigning an initial speed to each road segment of the road network, training the navigation map using training data including a training route for each of a multiplicity of trips by determining whether a number of trips for which a road segment is in the training route but not in the determined route is above a first threshold and, if the number of trips for which a road segment is in the training route but not in the determined route is above the first threshold, assigning a higher speed to the road segment, determining whether a number of trips for which a road segment is in the determined route but not in the training route is above a second threshold and, if the number of trips for which a road

segment is in the determined route but not in the training route is above the second threshold, assigning a lower speed to the road segment and determining a navigation profile including a route for a vehicle in the geographical area using the trained navigation map.

[0004] According to one embodiment, the first threshold and the second threshold depend on the number of trips for which the road segment is in the determined route, the training route or both.

[0005] According to one embodiment, the method includes determining whether the number of trips for which a road segment is in the training route but not in the determined route is above the first threshold and a third threshold and, if the number of trips for which a road segment is in the training route but not in the determined route is above the first threshold and the third threshold, assigning a higher speed to the road segment, and determining whether the number of trips for which a road segment is in the determined route but not in the training route is above a second threshold and a fourth threshold and, if the number of trips for which a road segment is in the determined route but not in the training route is above the second threshold and the fourth threshold, assigning a lower speed to the road segment,, wherein the first threshold and the second threshold are absolute thresholds and wherein the third threshold and the fourth threshold are relative thresholds depending on the number of trips for which the road segment is in the determined route, the training route or both.

[0006] According to one embodiment, initializing the navigation map includes assigning each road segment of the road network to a road segment speed group and iteratively training the navigation map includes for each road segment speed group, determining an average deviation between the routes determined using the navigation map and the training routes and tuning the speed of the road segment speed group to reduce the average deviation and, for each road segment of the road segment speed group, determining whether a number of trips for which a road segment is in the training route but not in the determined route is above a first threshold and assigning the road segment to a road segment speed group for a higher speed than the speed of the road segment speed group if the number of trips for which the road segment is in the training route but not in the determined route is above a first threshold and determining whether a number of trips for which a road segment is in the determined route but not in the training route is above a second threshold and assigning the road segment to a road segment speed group for a lower speed than the speed of the road segment speed

group if the number of trips for which the road segment is in the determined route but not in the training route is above a second threshold.

[0007] According to one embodiment, the method further includes, if neither the number of trips for which the road segment is in the training route but not in the determined route is above the first threshold nor a number of trips for which the road segment is in the determined route but not in the training route is above the second threshold, assigning the road segment to a road segment speed group having the same speed as the road segment speed group the road segment is from.

[0008] According to one embodiment, tuning the speed of the road segment speed group includes tuning the speed within a tuning range.

[0009] According to one embodiment, if the road segment speed group is a road segment speed group for a higher speed tuning the speed of the road segment speed group includes increasing the speed of the road segment speed group.

[0010] According to one embodiment, if the road segment speed group is a road segment speed group for a lower speed tuning the speed of the road segment speed group includes decreasing the speed of the road segment speed group.

[0011] According to one embodiment, the method includes generating a tuning tree having a node for each road segment speed group, wherein iteratively training the navigation map includes traversing the nodes of the tuning tree in a depth first search order and perform, when a node is being traversed, for the road segment speed group of the node, the tuning of the speed of the road segment speed group and, for each road segment of the road segment speed group of the node, the determining whether a number of trips for which a road segment is in the training route but not in the determined route is above a first threshold and the assigning the road segment to a road segment speed group for a higher speed than the speed of the road segment speed group if the number of trips for which the road segment is in the training route but not in the determined route is above a first threshold and the determining whether a number of trips for which a road segment is in the determined route but not in the training route is above a second threshold and the assigning the road segment to a road segment speed group for a lower speed than the speed of the road segment speed group if the number of trips for which the road segment is in the determined route but not in the training route is above a second threshold.

[0012] According to one embodiment, the method includes, when a node is being traversed, generating a child node for a road segment speed group for a higher speed and generating a child node for a road segment speed group for a lower speed except for leaf nodes of the tuning tree.

[0013] According to one embodiment, the method further includes evaluating the smoothness of the trained navigation map and re-training the navigation map if the smoothness has worsened by more than a predetermined maximum amount in the training or over a predetermined number of iterations of the training.

[0014] According to one embodiment, obtaining the road network for the geographical area includes obtaining information about the location of roads in the geographical area and subdividing the roads into the road segments.

[0015] According to one embodiment, the locations in the geographical area connected by the road segments are crossings of the roads and subdividing the roads into road segments includes subdividing the roads at the crossings.

[0016] According to one embodiment, the method further includes navigating one or more vehicles by the determined navigation profile.

[0017] According to one embodiment, the method further includes transmitting the determined navigation profile to a vehicle and controlling the vehicle to follow the route included in the navigation profile.

[0018] According to one embodiment, a server computer is provided including a radio interface, a memory interface and a processing unit configured to perform the method of any one of the above embodiments.

[0019] According to one embodiment, a computer program element is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of the above embodiments.

[0020] According to one embodiment, a computer-readable medium is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of the above embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

- FIG. 1 shows a communication arrangement for usage of an e-hailing service including a smartphone and a server.
- FIG. 2 illustrates a road segment speed tuning for one group of road segments.
- FIG. 3 illustrates a road segment speed tuning for a road network.
- FIG. 4 shows a flow diagram illustrating a method for according to an embodiment.
- FIG. 5 shows a server computer according to an embodiment.

DETAILED DESCRIPTION

[0022] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0023] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a vehicle or a method, and vice-versa.

[0024] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0025] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0026] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0027] In the following, embodiments will be described in detail.

[0028] An e-hailing app, typically used on a smartphone, allows its user to hail a taxi (or also a private driver) through his or her smartphone for a trip.

[0029] FIG. 1 shows a communication arrangement including a smartphone 100 and a server (computer) 106.

[0030] The smartphone 100 has a screen showing the graphical user interface (GUI) of an e-hailing app that the smartphone’s user has previously installed on his smartphone and has opened (i.e. started) to e-hail a ride (taxi or private driver).

[0031] The GUI 101 includes a map 102 of the vicinity of the user’s position (which the app may determine based on a location service, e.g. a GPS-based location service). Further, the GUI 101 includes a box for point of departure 103 (which may be set to the user’s present location obtained from location service) and a box for destination 104 which the user may touch to enter a destination (e.g. opening a list of possible destinations). There may also be a menu (not shown) allowing the user to select various options, e.g. how to pay (cash, credit card, credit balance of the e-hailing service). When the user has selected a destination and made any necessary option selections, he or she may touch a “find car” button 105 to initiate searching of a suitable car.

[0032] For this, the e-hailing app communicates with the server 106 of the e-hailing service via a radio connection. The server 106 includes a database 107 having information about the current location of registered vehicles 108, about when they are expected to be free, about traffic jams etc. From this, a processor 110 of the server 106 selects the most suitable vehicle (if available, i.e. if the request can be fulfilled) and provides an estimate of the time when the driver will be there to pick up the user, a price of the ride and how long it will take to get to the destination. The server communicates this back to the smartphone 100 and the smartphone 100 displays this information on the GUI 101. The user may then accept (i.e. book) by touching a corresponding button. If the user accepts, the server 106 informs the selected vehicle 108 (or, equivalently, its driver), i.e. the vehicle the server 106 has allocated for fulfilling the transport request.

[0033] It should be noted while the server 106 is described as a single server, its functionality, e.g. for providing an e-hailing service for a whole city, will in practical

application typically be provided by an arrangement of multiple server computers (e.g. implementing a cloud service). Accordingly, the functionality described in the following provided by the server 106 may be understood to be provided by an arrangement of servers or server computers.

[0034] To determine all route-related information, like the most suitable driver and an estimate of the time when the driver will be there to pick up the user, a price of the ride and how long it will take to get to the destination, the processor 110 access a data base 107 storing map data including for example one or more navigation maps 112 indicating where roads are located, the allowed direction of travel, speed limits, etc.

[0035] The data base 107 is in this example implemented by a local memory 109 of the server computer 106. However, it may also be implemented at least partially externally to the server computer 106, e.g. in a cloud, and it may be filled by access to an external data base 111, e.g. an open route map information database such as OSM (Open Street Map).

[0036] The quality of navigation of the services' vehicles 108 is of big significance for ride-hailing service. An accurate navigated route is critical for many location-based services such as calculating accurate ETA/ETT (estimated time of arrival/travel), pricing, order dispatch and multiple point travel integration. Meanwhile, it can improve drivers' and customers' experience significantly since better routes will save time, avoid additional cost, avoid traffic, avoid restriction and provide customized routes. Furthermore, efficient navigation of a fleet of vehicles avoids traffic jams, risk of accidents, reduces fuel consumption, etc.

[0037] The server 106 may determine the above information (such as ETA and ETT) by estimating the route that the driver will take. The server 106 calculates the route using a navigation map 112, stored in data base 107.

[0038] A navigation map 112 can be transformed into a big routing graph which consists of points and weighted lines. Based on all the weights in the graph, there are several shortest path determination techniques which the server 106 may use to determine a route for a trip (between a starting location and a destination) such as Dijkstra and contraction hierarchies to search and calculate routes from origin to destination.

[0039] However, a driver may not always go along the shortest or fastest path. There are many influential factors (e.g. traffic, toll, restriction, familiarity, weather, traffic light, turn and so on) that may affect a driver's choice. Commonly the influential factors of navigation

can be mapped into weights to tune the speed of a road or add time/cost penalties for the road (i.e. the edge of the routing graph). However, it is typically difficult to know every driver's preference and take all the actual local map conditions into account. So, according to one embodiment, the server 106 learns from historical data and tunes the speeds of roads of a navigation map 112. In other words, the server 106 uses historical data as training data for the navigation map 112. This allows the server 106 to estimate routes taken by drivers more accurately. The historical data includes routes taken by drivers and is stored in the data base 107. The route taken by a driver for a trip may be communicated from the driver's vehicle 108 to the server 106 (e.g. in form of a vehicle trace or driver trace, e.g. a GPS (Global Positioning System) trace) and the server 106 may store information specifying the route into the data base 107 to build up the historical data.

[0040] One approach for routing optimisation, i.e. for improving the estimation (or prediction or determination) of routes taken, or in other words, to improve the navigation map 112 in the sense that it better fits the actual geographical circumstances, is to learn the road type speed from historical data on road class level to learn, considering, for example, around 20 road classes such as motorway, trunk and so on (even if there may be millions of roads in a navigation map). The navigation map 112 may include a speed profile which specifies a speed for each road class, which means that the server 106 considers all the roads with the same class to have the same speed. The server 106 may tune the road class speeds for example by a grid search algorithm to improve route determination.

[0041] Although the routing optimisation at road class level can improve the prediction of routes taken (and similarly the navigation of vehicles in case the server 106 uses a determined route for navigating a vehicle 106), it is coarse-grained and the improvement is limited. Actually, not all the road's speed with the same road class is the same. For example, the speed of motorway in downtown and in suburb is typically different. Downtown motorway speed is slower than the speed in suburb in most conditions due to traffic.

[0042] Besides, the road-class level routing optimisation approach described above cannot address the issues of a road belonging to a private area road or being a toll road. In general, drivers would not or are not permitted to go through private areas. However, this is difficult to resolve by simply changing the speed in one road class as roads in private areas can have different road classes. Further, drivers would probably not choose toll roads if there is an alternative free route due to the extra cost of a toll road. In the same way, toll roads can have

different road classes so this is also difficult to resolve with the routing optimization approach described above.

In view of the above issues, according to various embodiments, a more fine-grained routing optimisation approach, working at a speed class level, is provided to solve the above problems. Speed class level means one road class speed can be split into several speed classes. According to various embodiments, roads are split into road segments and the speeds of the road segments are tuned. This allows treating different roads with different speeds. In contrast, the road class level optimisation approach described above cannot split the road speed into different road segments speeds because the grid search only tunes the road speed, which means all the road segments' speeds of the road will be tuned simultaneously. The road segments are the basic elements including a road. They are for example generated made by a map editor to represent a road. They are very short in common. For example, for some short distance from one crossing to the next there may be only one segment, but generally there may be two or more segments between crossings if there are other map marks, e.g. as observed in the map, e.g. an OSM map. Thus, road segments can not only be from crossing to crossing as nodes may not only refer to crossings but can bear other road attributes (such as traffic light, a barrier, etc). A segment can be considered as topologically connecting two nodes on the graph. For definition of nodes, the same concept as for OSM maps may be used.

[0043] According to one embodiment, to split a road speed into different road segment speeds, the server 106 uses a tuning tree to explore different speed classes of one road class using a depth first search order. Every tree node of the tuning tree has several road segments with the same speed and same road class. The server 106 generates three children nodes for the node with higher, equal and lower speed, respectively, following a statistics and classification strategy described below in detail. The server 106 allocates all the road segments to the children nodes independently. In the end, the server 106 generates different speed classes for road segments. This speed class level approach allows resolving the issues mentioned above for the road class level approach (mostly because grid search cannot scale well to road segments but a tuning tree can scale as it introduces speed classes). This allows improvement of route prediction (e.g. in terms of the offCourseRatio metric described below) for estimating routes drivers will take (e.g. for determination of ETA and ETT) or navigating (e.g. autonomous) vehicles or both. Furthermore, the speed class level approach generates classification information for each road class since it generates several speed classes for each

road class. This provides insight to understand the behaviour of each road class (e.g., probably for same road class, those in downtown area tends to have lower speed). Additionally, the speed class level approach can be easily applied and scaled to different cities.

[0044] As described in more detail below, the server computer 106 may further use a regularization term to monitor the tuning results in order to avoid overfitting.

[0045] As explained above, the routing optimisation approach applied by the server 106 according to various embodiments can be seen to focus on the speed class level. Given training data which may contain millions of booking traces (i.e. recorded routes for booked trips, also referred to as "bookings" in the following), an initial version of the navigation map 112 including a road network and, according to one embodiment, a speed profile which specifies a speed of each road class, the server 106 takes the speed of each road class, adjusts the speed of each road segment in depth-first search order and generates a tuning tree. It stores the results in a customized speed configuration file 113 containing information about the tuned road segments and the optimized speed associated with the road segments. The server 106 may use this speed configuration file 113 for routing (route prediction and navigation) that is more accurate than routing using the initial version of the navigation map (which does not contain the speed configuration file). It should be noted that the speed configuration file 113 is seen as part of the navigation map 112 (but may nevertheless be a separate file associated with the navigation map 112). In addition to the information from the speed configuration file 113, the navigation map 112 specifies a road network containing the road segments whose speeds are defined by the speed configuration file 113.

[0046] According to various embodiments, the routing optimization approach includes a metric called offCourseRatio. The OffCourseRatio measures how much a determined (i.e. a route plan) deviates from a driver trace (i.e. a route taken by a driver). The value of the OffCourseRatio ranges from 0 (completely same) to 1 (completely deviated). The offCourseRatio may be regularly checked during training of the speed classes. For large amounts of training data involving training in multiple tuning iterations, for example, the average value of offCourseRatio may be checked in each tuning iteration.

[0047] As mentioned above, inputs to the routing optimization are training data and an original navigation map (possibly with original road speed classes). Training data are historical routes taken by drivers (e.g. driver trace data) in the geographical region covered by

the navigation map. The navigation map is for example a navigation map for a city which allows determining a route plan of a trip based from the starting point and the destination point of the trip.

[0048] The server 106 initially calculates the original offCourseRatio (i.e. for the original navigation map). Then it generates a pseudo tree root and several child nodes which contain a set of map road segments separated by the original speed in descending order. Each road segment is related to the bookings that the driver trace covers that road segment.

[0049] Secondly, it starts to search at the first child node whose speed is the maximum as mentioned above. The search function is to use a strategy to perform statistics over the node's road segments and classify all the node's road segments into three subsets and mark them as three child nodes, e.g. also in speed descending order. The three subsets include a "higher", an "equal" and a "lower" subset which means the speed of a road segment moved to the next tree layer is assigned to the next layer subset depending on whether its speed is higher, equal or lower.

[0050] The server 106 then processes the next node according to depth first search. If the node's speed attribute is higher/lower, it increases/decreases the speed of the node's road segments by a tuning amount (within a tuning range) and updates the speed configuration file 113. Next, the server 106 uses the updated speed configuration file 113 to rebuild the navigation map 112, generate route plans for the training data (i.e. trips included in the training data) and calculates the resulting offCourseRatio. If the resulting offCourseRatio is lower than the one before tuning the speed, the server 106 updates it and writes the changed (i.e. tuned) speed to the speed configuration file 113 and stores it. After searching the tuning amount within the tuning range, which includes, for each tuning amount calculation of the resulting offCourseRatio and comparison with the resulting offCourseRatio of the previous tuning amount (starting from the untuned speeds), the server 106 gets a partially optimized road segment speed. It repeats performing statistics and classification to generate three child nodes in speed descending order and proceeds to the next tree layer continually.

[0051] In accordance with depth first search, when the down search reaches an end condition like a tree depth threshold or none of road segments being tuned etc., the server 106 stops proceeding to a deeper layer but continues processing with a brother node of the current node (i.e. the node processed last). If all its brother nodes have been processed, the search node returns to its parent's brother node on so on.

[0052] Thus, the server 106 generates a large tree whose nodes it has processed in depth first order. When this traverse process (including tuning the speeds at the nodes) is complete, the speed configuration file 113 holds the final result for the road segments speeds. After this process (which can be seen as training of the navigation map, specifically of the road segments speeds), in order to prevent overfitting, the server 106 may evaluate the smoothness of the generated navigation map with respect to the original map by a regularization term as described below. If the smoothness is not good, the server 106 may adjust input parameters (like the tuning range) and retrain the navigation map 112 until it gets better.

[0053] As a result of the navigation map training,

- Covered road segments are reassigned to a multi speed class level, rather than having a road type speed. The more training data there is available, the more accurate is the route prediction using the trained navigation map.
- The overall average offCourseRatio is guaranteed to be better than before training. Experiments show that it gets more than 15% improvement after tuning without overfitting in most cities using training data from driver traces of an e-hailing service. Thus, key OKR (objectives and key results) for an e-hailing service like ETT(estimated time of trip) and ETA(estimated time of arrival)/EDT(expected distance of trip) can be improved.
- Most private area roads and toll roads are probably reflected in the trained navigation map. Typically, the speed of private road segments will decrease and meanwhile the speed of road segments in the vicinity of the private roads will increase since the majority of drivers will not go through the private area and go around (e.g. parallel) to it. If most drivers do not choose toll roads despite the high speed, that toll road segments' speed will typically decrease to avoid being routed.

[0054] The navigation map training is adaptable such that it may be applied under different conditions and using different parameters, e.g.:

- Map type: city map, maps for routes for certain vehicles (e.g. 4-wheel maps, 2-wheel maps), nations, even customized maps.
- Time range of training data: all week time or just peak/non-peak hour.
- Tuning options: such as precision, tree depth, statistical decision conditions and so on.

- Filter road, initial road order: some road types may be filtered (e.g. filtered out) or different initial orders may be used instead of speed descending order.

[0055] Furthermore, the navigation map training may be easily scaled up (e.g. to a higher number of road types) and the tuning process may be intuitively interpreted.

[0056] In the following, one example of the road segment speed tuning process (i.e. navigation map training) is explained in detail.

[0057] For this, first, the representation of input and output data in this navigation map training is described.

1. Road Network (contained in navigation map 112)

   A road network is a weighted graph $G = (V, E, W)$, where the vertex set V consists of nodes, edge set $E \subseteq V \times V$consists road segments, each of which is represented by start node and end node with direction and W contains the weight function, which maps each road segment to a duration. Without loss of generality, other information in the road network (such as turn restrictions, one-ways, etc.) is ignored in the following as it does not affect the following explanation. The metadata of a road network, including node and road segment attributes (node location, road segment distance, road class of each road segment, etc) and topology information is stored in addition to the road network in the navigation map.

2. Route Plan

   A route plan $P =< v_1, v_2, \ldots, v_k >$is a sequence of nodes where two consecutive nodes are connected by an edge. An equivalent representation is $P =< e_1, e_2, \ldots, e_k >$where for any two consecutive road segments $e_i = (o_i, d_i)$ and $e_{i+1} = (o_{i+1}, d_{i+1})$ with $d_i = o_{i+1}$.

3. Driver's Trace

   A driver's trace is a time-ordered sequence of GPS pings capturing the movement of a vehicle 108, where a GPS ping is represented as a triplet (*ts, lat, lon*) captures the location (composed as latitude *lat* and longitude *lon*) of the vehicle at a timestamp *ts*. The distance between two locations is assumed to be haversine distance. The time gap between two consecutive pings is usually around one second.

4. Speed Profile

   A speed profile contains the mapping from road class to speed. In OSM (Open Street Map), each road segment contains a *road class* which is used to identify the type of

the road (e.g., motorway, primary road, etc.). The weight (duration) of each road segment can be easily derived from distance (the length between start node and end node) and speed (obtained from speed profile based on the road class it belongs to).

5. Training Data

   Training data D is used for tuning. It contains many historical booking (i.e. e-hailing trip) IDs, start and end points, their driver traces. An original route plan may be determined for each booking using the original (i.e. untrained) navigation map (i.e. using the road speed profile but not the road segment speeds determined in course of the training).

6. Configuration file

   The speed configuration file 113 is the resulting customized road segment level speed file. It is the result file of the navigation map training. It contains multi-road segments information lines. Each line consists of road segment's start point, end point and tuned speed. For determining a route, the server's route engine first loads the speed profile of the navigation map 112 to obtain all the road class speeds. Each road segment is initially given the road class speed in the speed profile of the road class to which the road segment belongs. After the training, the speed of the road segments are given by those in the speed configuration file 113. Thus, the routing speed of road segments of the same road class may have different values, which may generate better routing results.

7. Control Parameters

   Control parameters are a set of parameters controlling the training. They for example contain training data time range, tuning tree depth, tuning precision, road class whitelist, tuning maximum speed, tuning step, statistical distribution percentage threshold, minimum bookings, map city, map wheels, map version, phase (pickup or drop-off) and so on. They allow to adapt the training to different cities, different time and different training requests.

[0058] The "OffCourseRatio" is the metric to measure how much a determined route plan (i.e. a determined route) P deviates from the driver's trace $\hat{P}$. Given a driver's trace $P =< e_1, e_2, \ldots, e_k >$ and route plan $\hat{P} =< \hat{e}_1, \hat{e}_2, \ldots, \hat{e}_k >$, offCourseRatio can be defined as

$$OffCourseRatio(\hat{P}, P) = 1 - Sim(\hat{P}, P).$$

with

$$Sim(\hat{P}, P) = \frac{SequentialOverlap\ (\hat{P}, P)}{\sum_{e \in \hat{P}} \quad len(e)}$$

being the similarity of the route plan and the driver's trace, which is measured by Jaccard similarity index. len(e) denotes the length of edge e. An exemplary implementation of $SequentialOverlap$ is described in the following Algorithm 1.

Algorithm 1: SequentialOverlap

Input: $\hat{P}$, P
Output: Length of Overlap

```
i, j ← 0 // match P̂(i) with P(j) starting with the first element
match ← 0 // length of match
while i < size(P̂) and j < size(P) do
    // find the first appearance of P(j) in P̂ with index no smaller than i
    m ←FirstMatch(P̂, i, P(j))
    if m >= i then
        match ← match + len(P(j))
        i ← m + 1
    j ← j + 1
return match
```

[0059] The more edges a determined route shares with driver's trace, the more similar the two paths are. The overlap of two paths is required to be sequential as the sequence of edges in route plan is connected in a timely order.

[0060] As can be seen, the offCourseRatio lies between 0 and 1. A value of 0 means that the two routes are completely the same while 1 means that there are totally different. An offCourseRatio value may be determined for every booking in the training data set D using the current version of the navigation map (starting with the untrained version and proceeding with a more and more fully trained navigation map). Each booking is independent and the navigation map is trained to optimize the average value of the offCourseRatio over the training data.

[0061] This means that the optimization problem solved by the training is

$$Min \frac{\sum_{i \in D} \quad offCourseRatio(i)}{N}$$

$s.t.$

$$D \ = \ training\ dataset$$

$$N = numbers\ of\ bookings\ in\ D$$

$$offCourseRatio(i) = 1 - \frac{SequencialOverlap(P_i, P_i')}{\sum_{e \in P} len(e)}$$

$$P_i = < e_1, e_2, \ldots, e_k >_i \ // driver's\ trace$$

$$P_i' = < e_1, e_2, \ldots, e_k >_i' // route\ plan, determined\ by\ speed$$

This means that the goal is to minimize the input training dataset's average offCourseRatio. The offCourseRatio is affected by the route plan calculated by server's route engine. The route engine is configured by the speed profile and the customized speed configuration file 113. In course of the training, the road segment speeds are repeatedly tuned and each time, for a current tuning the route engine is started to calculate a route plan from each booking's start and end point for the current tuning. Then the server calculates the offCourseRatio between the determined route plans and the routes from the training data and compares whether the current tuning has improved the offCourseRatio. If it has improved, it keeps the tuned road segment speeds. Otherwise, it reverts to the previous values (before current tuning).

[0062] As described above, the server 106 performs statistics (i.e. determines statistical information) and applies depth first search in a tree data structure and classification of the road class speed into several new speed classes.

[0063] In the tree data structure, each node (except the leaves) has three children nodes, one for each speed class "higher speed class", "equal speed class" and "lower speed class". Table 1 shows a classification strategy for classifying each road segment into one of those speed classes.

Classification Strategy For each road segment		Route Engine Route	
		Yes	No
Driver's Trace	Yes	YY -> speed is accurate	YN -> need to increase
	No	NY -> need to decrease	None

Table 1

[0064] It should be noted that the server 106 performs the navigation map training under the assumption that the drivers which have chosen the routes of the training data have always chosen the fastest route they think.

[0065] As shown in table 1, for each road segment occurring in context of a booking there are three situations.

[0066] YY: If the road segment is both in a booking's route plan (determined by the route engine) and the driver's trace of the booking (from the training data), the speed is accurate for this road segment (with respect to this booking).

[0067] YN: If the road segment is in a driver's trace of a booking but not in a determined route plan for the booking, it means the current speed is slow so that route engine did not include this road segment in the route. So, this is an indication that the speed should be increased.

[0068] NY: If the road segment is in a determined route plan for a booking but not in the driver's trace for the booking, it means current speed is quick so that route engine routed to this road segment but driver did not choose it actually. So, this is an indication that the speed of the road segment should be decreased to prevent the route engine to route to this road segment.

[0069] Usually, a road segment will occur in multiple bookings (i.e. determined routes or driver trace or both of multiple bookings. Therefore, the server 106 performs statistics over the three situations YY, YN, NY of table 1, e.g. counts the numbers of bookings #YY, #YN and #NY for which YY, YN and NY, respectively, and compare the proportion to determine whether to increase the road segment's speed, keep it equal or decrease it. For example, the decision criteria to classify a road segment to a speed class may be as follows.

1. $\#YN/(\#YY + \#YN + \#NY) > 0.7 \; and \; \#YN > threshold$ if YN proportion is above 70% and the absolute number is above a predetermined (e.g. input) threshold like 100, it means this road segment speed should increase in the majority of bookings. So it will allocate the road segment into higher speed class;
2. $\#NY/(\#YY + \#YN + \#NY) > 0.7 \; and \; NY > threshold$ if NY proportion is above 70% and the absolute number is a above a predetermined (e.g. input) threshold like 100, it means this road segment speed should decrease in the majority of bookings. So it will allocate the road segment into lower speed class;

3. Otherwise the road segment will be allocated to equal speed class, which means those road segments speed does not need to change in this node classification.

[0070] It should be noted that the inequalities with thresholds 0.7 for the relative numbers may also be seen as inequalities for the absolute numbers #YN and #NY with thresholds which depend on #YY+#YN+#NY by multiplying both sides with #YY+#YN+#NY (resulting in thresholds 0.7 * #YY+#YN+#NY for the absolute numbers).

[0071] An exemplary implementation of the classification is described in the following Algorithm 2.

Algorithm 2 Statistics and Classification in one node

Input: **node**=current tuning node, **minB**=absolute minimum booking numbers

Output: **lowNode,equalNode,highNode**=3 children speed classes node

New $lowNode$, $equalNode$, $highNode$, mark $tuneDirection$ as 'low','equal','high' respectively
$node.children = [lowNode, equalNode, highNode]$
for each $segment$ in $node.set$ **do**
    $equalNums = segment.routeBookings \cap segment.traceBookings$ // segment in both driver's trace and route plan
    $highNums = segment.traceBookings - segment.routeBookings$ // segment in driver's trace not in route plan
    $lowNums = segment.routeBookings - segment.traceBookings$ // segment in route plan not in driver's trace
    $total = equalNums + highNums + lowNums$
    if $highNums/total > 0.7$ and $highNums > minB$ **then**
        $highNode.set.Add(segment)$
    else if $lowNums/total > 0.7$ and $lowNums > minB$ **then**
        $lowNode.set.Add(segment)$
    else
        $equalNode.set.Add(segment)$
    end if
end for
return $lowNode, equalNode, highNode$

[0072] In Algorithm 2, the operation Add(road segment) adds the road segment to a respective child node.

[0073] FIG. 2 illustrates the road segment speed tuning performed for one node 200 of the tuning tree. At the start of the processing, the node contains (i.e. is associated with) a set of road segments 204 with the same (initial) speed.

[0074] Three children nodes 201, 202, 203 are defined for the node 200: a higher speed class node 201, an equal speed class node 202 and a lower speed class node 203. Each of the node 200 and the children nodes 201, 202, 203 has an identification (ID).

[0075] The speed of the node (i.e. of the set road segments 204) is tuned in 205.

[0076] For the tuning, the server 106 starts a loop with speed tuning range [x,y] (e.g., [x,y] is [-2,t]/[0,0]/[-t,2] for tuning direction “high”/”equal”/”low”; 2 or -2 is just for safeguarding; tuning direction “equal” as well as “original” means that no tuning is performed). Over the iterations of the loop, starting from the nodes initial speed, the speed of all the road segments 204 traverses the speed tuning range with respect to the original speed in the tuning direction (“high” or “low”). In each iteration of the loop, the server 106 updates the speed configuration file, starts the route engine to get a determined route plan based on the booking’s start and end point, calculates the offCourseRatio in 208 (using the driver’s trace 207 from the training data of the booking) and compares with the previous offCourseRatio in 209 (i.e. the offCourseRatio which results for the speed without the tuning of the present iteration). If the offCourseRatio has improved, the tuned speed of the present iteration is kept as the node’s speed (possibly to be further changed in the following iterations of the loop). As a result, after all iterations, it will get an optimized speed 210 of the node’s road segments 204.

[0077] In 211, after getting the optimized speed 210, the server 106 performs statistics and classifies the road segments 204 by the classification strategy described above with reference to table 1 and algorithm 2 into the three speed classes of the three children nodes 201, 202, 203. The server 106 gives the three children nodes the optimized speed 210 as initial speed and gives the tuning direction “high” to the higher speed class node 201, “equal” to the equal speed class node 202 and “low” to the lower speed class node.

[0078] An exemplary implementation of the speed tuning of a node is described in the following Algorithm 3.

Algorithm 3 OneNodeTuning

Input: **node**=tuning node containing set of segments, **D**=train data; **map**, **step** = tune step, **precision**=tuning offCourse ratio precision, **S**=iterated output file speed.csv, **M**=iterated offCourse ratio, **R**=iterated route plan,

Output: **S,M,R**

$S_{ori} = S$, $M_{ori} = M$, $R_{ori} = R$, $Speed_{ori} = node.speed$
for δ in range [-2, *step*] **do**
  if *node.tuneDirection* == 'high' **then**
    $node.speed \mathrel{+}= \delta$
  end if
  if *node.tuneDirection* == 'low' **then**
    $node.speed \mathrel{-}= \delta$
  end if
  if $node.speed > maxS$ or $node.speed <= 0$ **then**
    Continue // ignore exceeds speed
  end if
  S_{tmp} = UpdateSpeedCsv(*node*) // update speed.csv with all the segments in the node with current node.speed
  R_{tmp} = StartRouteEngineAndGetRoutePlan(*D*,*map*,S_{tmp})
  M_{tmp} = CalOffCourseRatio(R_{tmp}, *D*)
  if M - $M_{tmp} > precision$ **then**
    $M = M_{tmp}$; $R = R_{tmp}$; $S = S_{tmp}$
  end if
end for
// if offCourse ratio improvement is not enough, reset arguments
if $M_{ori} - M <= precision$ **then**
  $S = S_{ori}$, $M = M_{ori}$, $R = R_{ori}$, $node.speed = Speed_{ori}$
end if
return S, M, R

[0079] FIG. 3 illustrates the road segment speed tuning performed over the whole tuning tree 300.

[0080] In this example, the tuning tree 300 has, in a first layer, a pseudo root 301 (not associated with any road segments) and a second layer of nodes 302, 303, 304 and further layers, wherein, as explained above, starting from the second layer each node has three children nodes for each speed class (high, equal and low). In figure 3, children nodes 305, 306, 307 of the third layer, which are the children nodes of the leftmost node 302 of the

second layer, are shown. Further, children nodes 308, 309, 310 of the fourth layer, which are the children nodes of the leftmost node 304 of the third layer, are shown.

[0081] The server 106 performs speed tuning over the nodes 302-310 from the root to the leaves (in a depth first search order, e.g. pre-order traversal).

[0082] According to one embodiment, on each layer, the nodes are ordered such that their speed descends from left to right.

[0083] The nodes 302, 303, 304 of the second layer are associated with the road classes, i.e. each node 302, 303, 304 of the second layer is given all road segments of roads of a certain road class. Accordingly, the server 106 does not perform speed tuning for the second layer nodes 302, 303, 304 because their speed is the road class speed as in the speed profile of the road classes. It is assumed that highways have the highest speed and influence routing the most.

[0084] Every tree node's initial speed and initial offCourseRatio is given by on its parent's optimized speed and the offCourseRatio corresponding to this optimized speed. So, once the whole tree 300 has been traversed, the result is a final offCourseRatio and road segments speed classes of all road segments, optimized for the input training data.

[0085] For equal speed class nodes 306, 309, the server performs statistics and classification but does not tune the speed because this is covered by the tuning of the parent node's speed.

[0086] There are for example three end conditions of the depth first search in the tuning tree, i.e. a node is a leaf when:

1. None of the road segments of the node is allocated into higher/lower speed class. So there's no need to tune.
2. A tuning tree depth threshold (which is e.g. an input parameter) has been reached.
3. The node tune direction is "equal" and its parent is also an "equal" speed class node, which means there is no need to perform statistics and classify the road segments because the parent node does not change its speed so it will not affect the later nodes.

[0087] When the whole tree 300 has been traversed, every leaf node represents a speed class. The server 106 generates a speed configuration file 113 which contains every tuned road segment information and speed. With this speed configuration file, the server 106 can (by means of its route engine) predict routes more accurately. The offCourseRatio is optimized for the training data.

[0088] An exemplary implementation of the speed class level routing optimization is described in the following Algorithm 4. It uses StatisticsAndClassification of Algorithm 2 and OneNodeTuning of Algorithm 3.

Algorithm 4 Routing Optimisation at Speed Class Level

Input: speed profile, control parameters(**T** = time range of data, **minB** = minimum bookings, **L** = whileList, **city** = city, **wheel** = wheels, **mapversion** = map version, **phase** = pickup or dropoff, **dep** = tree depth threshold, **step** = tune step, **maxS** = max speed, **precision** = offCourse ratio precision

Output: **S**:speed.csv (contains tuned segments speed)

// Get original data from input parameters
D = DownloadTrainingData(T, city, wheel, phase) // training data from cloud storage
map = DownloadMap(speed profile, mapversion) // map file
S = InitialSpeedCsv() // initialize output file, nil at begin
R = StartRouteEngineAndGetRoutePlan(D, map, S) // original route plan
M = CalOffCourseRatio(R, D) // original offCourse ratio
// Initialize segment speed tree and stack
New *stack*, tree pseudo *root*
Classify each input booking's segments to tree node based on speed in ascending order, mark $tuneDirection$ = 'original'
Make all the above tree nodes into its child
$stack.Push(root, root.children)$
// Start DFS
while stack is not none **do**
    $node$ = $stack$.pop()
    if $node.tuneDirection$ == 'original' or 'equal' **then**
        do nothing
    else if $node.tuneDirection$ == 'high' or 'low' **then**
        // In every node tuning, it will update M, R, S
        M, R, S=**OneNodeTuning**($step, D, map, tuneDirection, M, R, S, precision$)
    end if
    $depth$ = GetCurrentDepthInTree($node$)
    // end conditions of searching down
    if $depth$ == dep **then**
        Continue
    else if $node.tuneDirection$ == 'equal' and $node.parent$!= nil and $node.parent.tuneDirection$ == 'equal' **then**
        Continue
    end if
    // Statistics and Classification, it will generate 3 children nodes
    $lowNode, equalNode, highNode$ = **StatisticsAndClassification**($node, minB$)
    $stack.Push(lowNode, equalNode, highNode)$
end while
return S

[0089] By the above process, the road class speed is split into several road segment speed classes after tuning. There is a risk that navigation using these speed classes overfits, which

means a determined route may not be smooth which results in that drivers may have to take turns a lot of times. So in order to avoid overfitting, according to one embodiment, a regularization term is used which measures the smoothness:

$$smoothness\ factor\ = \sum_{(i,j)\leftrightarrow(k,l)} \left|\frac{1}{v_{(i,j)}} - \frac{1}{v_{(k,l)}}\right| \frac{2}{d_{(i,j)} + d_{(k,l)}}$$

[0090] As the formula showed above, two road segments (i,j) and (k,l) are called neighboring when they represent consecutive or intersecting roads. The neighbouring relationship is written as $(i,j) \leftrightarrow (k,l)$. $v_{(i,j)}$ corresponds to the speed in kilometers per hour of the road segment (i,j). $d_{(i,j)}$ represents the length in meters of the road segment (i,j). $\sum_{(i,j)\leftrightarrow(k,l)}$ represent the sum over all pairs of neighbouring road segments (i,j) and (k,l) in the road segments occurring in the training dataset. The weighting factor $2/(d_{(i,j)} + d_{(k,l)})$ ensuring that continuity is more important in shorter neighbouring roads (where constant velocity is a better approximation) than in longer ones. In other words, the smaller the smoothness factor is, the smoother the navigation map is.

[0091] If the smoothness factor becomes worse after speed tuning (of some nodes or after performing the complete route optimization), the server 106 may adjust the controlling parameters and re-tune until it gets better. Thus, it can avoid overfitting while optimizing the road network navigation.

[0092] In summary, according to various embodiments, a method is provided as illustrated in FIG. 4.

[0093] FIG. 4 shows a flow diagram 400 illustrating a method for determining a navigation profile for a vehicle in a geographical area.

[0094] In 401, a road network is obtained for the geographical area, the road network including road segments connecting locations in the geographical area.

[0095] In 402, a navigation map is initialized by assigning an initial speed to each road segment of the road network (wherein the initial speed is for example given by the speed of a road class of a road of which the road segment is part).

[0096] In 403, the navigation map is trained using training data including a training route for each of a multiplicity of trips by,

Determining whether a number of trips for which a road segment is in the training route but not in the determined route is above a first threshold and, if the number of trips for which a road segment is in the training route but not in the determined route is above the first threshold, assigning a higher speed to the road segment and

Determining whether a number of trips for which a road segment is in the determined route but not in the training route is above a second threshold and, if the number of trips for which a road segment is in the determined route but not in the training route is above the second threshold, assigning a lower speed to the road segment.

[0097] In 404, a navigation profile including a route for a vehicle in the geographical area using the trained navigation map is determined.

[0098] According to various embodiments, in other words, the speed or road segments which are seldomly taken by drivers is decreased while the speed of road segments which are often taken by drivers are increased. Thus, road segment speed is used as a parameter indicating how suitable (or popular) a road segment is for a trip. Having trained these road segments speeds allows efficient navigation through the geographical area. Training the navigation map may be seen as learning from demonstrations, i.e. training a machine learning model from demonstrations, wherein in this example the demonstrations are routes taken by drivers and the machine learning model is trained to come up with routes for trips itself. It may determine the route of a trip from the trained navigation map by means of a shortest-path algorithm which takes into account road segment speeds (i.e. an algorithm which calculates a shortest path between vertices taking into account edge weights). So, the machine learning model that is trained can be seen to include the navigation map and the route engine and it is trained to come up with similar routes as the ones of the training data (and to generalize to unseen trips).

[0099] Thus, the method of FIG. 4 may be part of a method for controlling a vehicle (or controlling the navigation of a vehicle) wherein the trained navigation data is provided to a vehicle and the vehicle (or its navigation system) is controlled using the trained navigation map data.

[00100] For example, the route to take for a trip may be determined by the server 106 and transmitted to the respective vehicle 108, i.e. the vehicle assigned to the trip, such that the driver may follow the route. In case of an autonomous vehicle the vehicle may be controlled

according to the determined route. The server 106 calculates the route using a navigation map, stored in data base 107.

[00101] The term navigation profile may refer to a route given to a vehicle for navigating the vehicle or to the prediction of a route the vehicle will take.

[00102] The various data like the navigation map, the road network and the training data are electronic data and the method may be computer-implemented.

[00103] The method of FIG. 4 is for example carried out by a server computer as illustrated in FIG. 5.

[00104] FIG. 5 shows a server computer 500 according to an embodiment.

[00105] The server computer 500 includes a communication interface 501 (e.g. configured to receive drivers' traces or the specification of a road network or to transmit navigation map data or navigation profiles to another server (e.g. a navigation server) or a navigation device in a vehicle). The server computer 500 further includes a processing unit 502 and a memory 503. The memory 503 may be used by the processing unit 502 to store, for example, driver's traces and a road network to be processed. The server computer is configured to perform the method of FIG. 4.

[00106] The methods described herein may be performed and the various processing or computation units and devices described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[00107] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A method for splitting a road speed into different road segment speeds, the method comprising:

exploring different speed classes of a road class using a depth first search order,

wherein the exploring different speed classes comprises using a tuning tree including a plurality of tree nodes, and each node of the plurality of tree nodes includes a plurality of road segments;

generating three children nodes based on speed; and

allocating each of the plurality of road segments to the generated children nodes, so that different speed classes for the road segments are generated.