Inferring Waypoints in the Absence of Knowledge of Driving Style

Daniel A. Desmond, Kenneth N. Brown

Insight Centre for Data Analytics, Department of Computer Science, University College Cork, Cork, Ireland
{daniel.desmond, ken.brown}@insight-centre.org

Abstract. We present an algorithm for predicting intervals which contain waypoints from a GPS trace of a multi-part trip without having access to historical data about the driver or any other aggregated data sets. We assume the driver's driving style is not known, but that it can be approximated by one of a set of cost preferences. The method uses a set of repeated forward and backward searches along the trace, where each of the searches represents one of the driving costs. We evaluate the algorithm empirically on multi-part trips on real route maps. The algorithm selects the results of the search with the fewest number of intervals and we achieve over 95% recall on estimating waypoints while the intervals cover less than 9% of the trace.

1 Introduction

Recreating intent from activity traces is an important aspect of security, missing persons search, assisted living, and retail analysis. In the retail analysis case the inference would require the analysis of large sets of activity traces from many people, while in assisted living there will be many traces from one person. In the security and missing persons case the inferences may have to be determined from a single trace for an individual. In such cases the activity traces may be trajectories through space with intermediate waypoints intepreted as intent.

In previous work [1] we showed it was possible to predict intervals which contain up to 97% of waypoints from a driver's GPS trace when a driver's route choice is based on shortest paths. We now we relax this assumption and consider the case where the driver's route choice may be based on different criteria. We assume that we have a model which is an abstraction of road network, which contains details of travel times and distances, tolls and road types but no other information. We present an algorithm which compares the results of a series of repeated forward and backward searches for shortest paths based on different cost functions which approximate different route choices. We evaluate the algorithm empirically using randomly generated locations from which multi-trip routes are generated by an online route planner. We demonstrate that the algorithm generates intervals in the traces which cover over 95% of the waypoints, and where the intervals cover less than 9% of the trace.

The remainder of the paper is organised as follows: Section 2 discusses related work. Our proposed approach to the problem is introduced in section 3. Section 4 describes the form of the experiments. The results of the experiments are reported in section 5 and section 6 concludes the paper.

2 Related Work

Wardrop [2] proposed that the minimization of travel time was the most important criterion in route selection and his first principle of route choice states "the journey times on all routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route". Studies by Duffel and Kalombatis [3] and Huchingson et al. [4] also found that travel time was considered to be the most important factor when deciding on which route to take. However in a more recent review of multiple studies Chen et al. [5] showed that route choice was based on more than travel time, with the driver's preferences, familiarity with the areas being travelled and other judgements also accounting for how routes were chosen. Zhu and Levinson [6] also carried out an empirical test of Wardrops's first principle and found that the majority of people did not choose the shortest path by time.

Currently the majority of research based on prediction or inference using GPS traces revolves around destination prediction and establishing if patterns exist such as popular or heavily travelled routes in the traces being examined. These require the use of a number of different types of machine learning methods such as pattern recognition [7] and Hidden Markov Models [8] among others. All of these methods require the use of large amounts of historical data to build up their models and do not use a graph of the geographical area being studied. Where the inference of waypoints via creating sub-traces from the trace is used, the ultimate aim is still to predict the destination [9]. The most similar problem to the one we are tackling here is one studied by Kafsi et al [10], where the aim is to infer a set of waypoints from a GPS trace. They ignore the time component when segmenting the trace but use historical data to estimate the waypoints computing the entropy of conditional Markov trajectories. To the best of our knowledge, we are the first to present a method for inferring waypoints where historical data is not required and we only assume that the driver is consistent in their driving style for the whole trip.

There exist a multitude of trip planners both on-line and off-line such as Google maps [11], and those built using openstreetmap data(OSM)[12] data [13,14]. Openrouteservice [15] is an open source route planner in which it is possible to create multi-point trips with different travel profiles or cost functions and download the trace data for the trip.

In previous work [1], we demonstrated a method for inferring waypoints using only a GPS trace and no other historical or aggregated data, where the assumption was made that the creator of the trace used shortest paths by time. The method is based on repeated forward and backwards searches using Dijkstra's algorithm [16]. The method is discussed in section 3.

3 Approach

Our hypotheses are

- Given a multipart trajectory constructed from point-to-point trips, in an unknown driving style, the individual destinations (waypoints) can be inferred from the results of multiple simulations of different driving styles.
- The simulated driving style which produces the fewest **waypoint** estimates generates a reliable estimate of the true waypoints.

Let $G = (V, E, t, d, c, H, h)$ be a strongly connected, multi-weighted, directed graph embedded in a two-dimensional (2D) space. V is the set of vertices where each vertex is a location in the space. E is the set of directed edges (v_i, v_j) where $v_i, v_j \in V$ and so each edge represents a road segment. t is a function $t : E \longrightarrow \mathbb{N}^+$ representing the cost of traversing an edge in seconds. d is a function $d : E \longrightarrow \mathbb{N}^+$ representing the cost of traversing an edge in miles. c is a function $c : E \longrightarrow \mathbb{N}^+$ representing the cost of using a toll in US dollars. h is a function $h : E \longrightarrow H$ representing the type of road of the edge. H is set of different road types.

A trip s is a sequence of vertices and $\bar{s}$ is the last vertex in s. A multitrip M is a sequence of trips $\langle s_1, s_2, ..., s_j \rangle$ such that $\bar{s_i}$ is the first vertex in s_{i+1}. A flattened multitrip is a sequence of vertices created by flattening the multitrip. A trace $T = \langle v_1, v_2, ..., v_k \rangle$ is a sequence of vertices sampled in order from the flattened multitrip. Given a trace our aim is to reconstruct the individual trips i.e. the endpoints $\langle \bar{s_1}, \bar{s_2}, ..., \bar{s_{j-1}} \rangle$ from the multitrip. We allow a relaxation in which our aim is to output a list of intervals $\langle [a_1, b_1], [a_2, b_2], ..., [a_j, b_j] \rangle$ where $\bar{s_i}$ is contained within $[a_i, b_i]$.

For this work we define a driving style to be a route choice preference function using an appropriate cost function χ. A shortest χ-path is the path with the smallest χ cost. If χ is based on travel time then the χ least cost path will be the path that takes the least time. To accommodate inaccuracies in travel costs in our abstraction of the road network and the driver's bounded knowledge of the route taken we introduce an ε-shortest χ-path.

Definition 1. *ε-shortest χ-path:* Path P from A to B is an ε-shortest χ-path from A to B if there is no other A, B path Q with $\chi(\mathrm{Q}) \leq \chi(\mathrm{P})$ - ε.

Definition 2. *ε-shortest χ-path(percentage):* Path P from A to B is an ε-shortest χ-path from A to B if there is no other A, B path Q with $\chi(\mathrm{Q}) \leq (\frac{100-\varepsilon}{100}) * \chi(\mathrm{P})$, where ε is a percentage.

In our previous work [1] we presented an algorithm and demonstrated its success at inferring intervals containing waypoints when the driver was following the shortest path by time. In the current work we do not know the driving style. Instead we simulate different driving styles (approximated by different cost functions χ) and infer the intervals which contain waypoints from the results of the computations. The new algorithm modified for a specific cost function χ is

Algorithm 1: Waypoint Estimation With Cost Function

input : Allowable Tolerance ε
input : Heading Tolerance α
input : cost function χ
input : Trace T
output: List K of intervals

List K
List ST `// will hold the calculated sub-traces`
$subTraceStart \longleftarrow T[0]$
for $i \leftarrow 2$ **to** *last point in* T **do**
    $previous \longleftarrow T[i-2]$
    $current \longleftarrow T[i-1]$
    $next \longleftarrow T[i]$
    $heading1 \longleftarrow$ heading traveled from *previous* to *current*
    $heading2 \longleftarrow$ heading traveled from *current* to *next*
    if *difference between heading1 and heading2 is an* α*-heading change* **then**
        add interval [*previous*, *next*] to K
        add sub-trace from *subTraceStart* to *previous* to ST
        $subTraceStart \longleftarrow next$
    end
end
for $st \in ST$ **do**
    $source \longleftarrow st[0]$
    while *not at end of st* **do**
        Searching from source find first point B on st which is not a χ-cost ε-shortest path
        Searching back from B find first point A on reverse search of st which is not a χ-cost ε-shortest path
        add interval [A,B] to K
        $source \longleftarrow B$
    end
end
return K

shown in Algorithm 1. The algorithm infers waypoints using two methods which are carried out sequentially. In the first method (lines 4-15) we identify abrupt reversals of direction which would be caused by driving in to an area and leaving by the reverse route and add these to the intervals. As each point is a location in 2D space, each successive pair of points in trace T has a direction between them. Therefore we define a α-heading change as follows

Definition 3. *α-heading change:* Difference between heading of travel from t_{i-1} to t_i and heading of travel from t_i to t_{i+1} is $180^\circ \pm \alpha^\circ$.

We then split the trace into subtraces, separated by the intervals for abrupt reversals, where each subtrace is made up of contiguous points of the trace. The second method (lines 16-24) for inferring waypoints involves searching forward through a subtrace until we find a point B which is the first point not on a ε-shortest χ-path, then search backwards from B until we find point A which is the first point not on a ε-shortest χ-path from B. Interval [A,B] is added to the list of intervals, and the forward search resumes from point B. When we have reached the end of the subtraces we return the list of intervals found.

Based on the results of [1] a search that uses the correct cost function should produce approximately the correct number of intervals (or fewer if one waypoint is on the shortest path between its predecessor and successor waypoints). A simulation using an incorrect cost function χ should introduce extra intervals to account for deviations from its shortest χ-path, although it is possible that it could miss a waypoint between the intervals for the same reason as above. Therefore we suspect that the cost function that returns the fewest intervals is a good estimate for the true driving style. If more than one function produces the fewest intervals we need a tiebreaker. Given a list of intervals, we assume each interval will need to be examined for places of interest, therefore we could choose wide intervals where we are more likely to find waypoints, which we call *maximum spread*, or we could choose narrow intervals where the search time would be smaller at the cost of missing a waypoint, which we call *minimum spread*, where spread is defined as total width of the intervals in seconds. We define the spread in terms of seconds since the GPS points are timestamped and we cannot compute distance travelled between GPS points. The results from both of these options will be compared in section 5. If there is still more than one function remaining then random choice will be used to select one. Other options such as selecting by the maximum spread or minimum spread initially and then using the number of estimates as a tiebreaker were implemented but gave poor results and so are not discussed further.

The pseudocode for the Waypoint Estimation Assuming No Driving Style Information is shown in Algorithm 2. The inputs are the trace T, ε, α, and a set of cost functions representing different driving styles. Initialize two lists, AK which will hold the list of each the intervals returned from each of the waypoint estimation algorithms we are interested in and K to hold the intervals which will be returned(lines 1-2). A counter C to keep track of the minimum number of intervals returned by any cost function is initialized to ∞ (line 3). For each of the

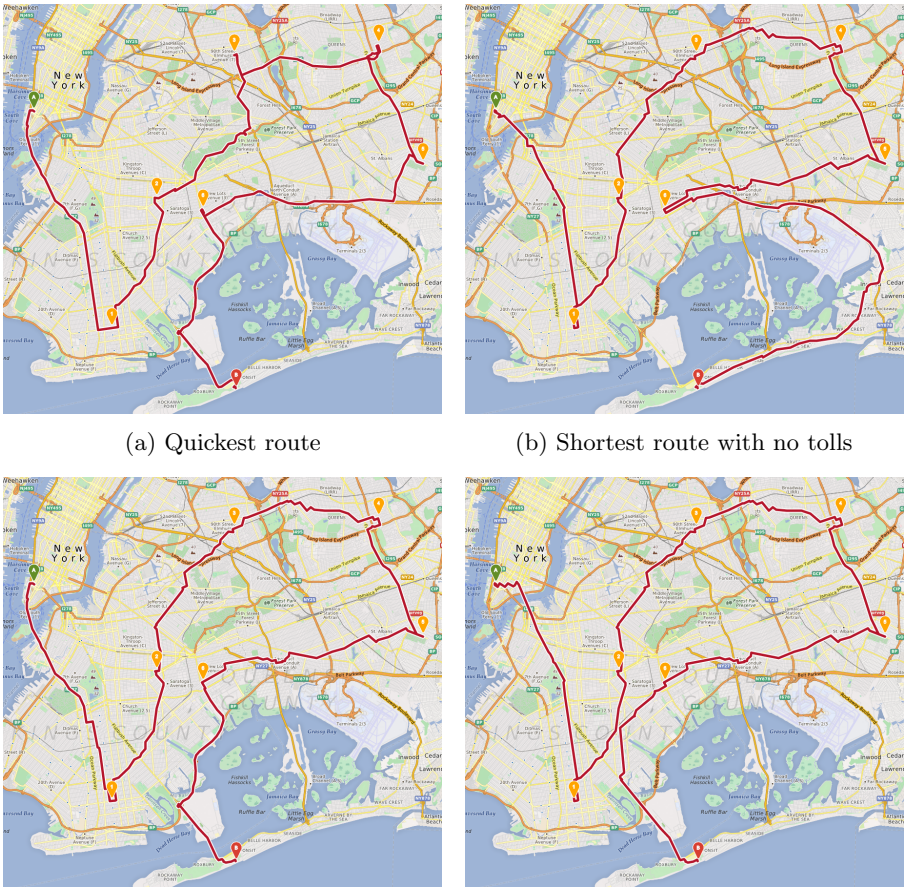
cost functions we return a list L of intervals (line 5). If the number of intervals is equal to C then L is added to AK (lines 6-8). If the number of intervals is less than C then AK is cleared, L added to the now empty list and C set to the number of intervals in L (lines 9-13). If AK contains only one list of intervals then set K equal to this list of intervals (line 16). If AK contains more than one list, we select the one with minimum or maximum spread. If there are still ties, we randomly select a list with minimum or maximum spread (lines 17-24). Return the list K of the intervals found (line 25).

Algorithm 2: Waypoint Estimation Assuming No Driving Style Information

```
   input  : Allowable Tolerance ε
   input  : Heading Tolerance α
   input  : Set of Cost functions X
   input  : Trace T
   output: List K of intervals
 1 List AK
 2 List K
 3 Integer C ⟵ ∞
 4 for χ ∈ X do
 5 |   List L = Waypoint Estimation with Cost Function (T, ε, α, χ)
 6 |   if L.size = C then
 7 |   |   add L to AK
 8 |   end
 9 |   if L.size <C then
10 |   |   clear AK
11 |   |   add L to AK
12 |   |   C ⟵ L.size
13 |   end
14 end
15 if size of AK = 1 then
16 |   K ⟵ the only list of estimates in AK
17 else
18 |   List S = Select From AK based on either the minimum or maximum
   |     spread of the intervals in each list
19 |   if size of S = 1 then
20 |   |   K ⟵ the only list of intervals in S
21 |   else
22 |   |   K ⟵ list of intervals selected randomly from S
23 |   end
24 end
25 return K
```

4 Experiments

To evaluate the algorithms we simulate routes in the city of New York. The algorithms were implemented in Java 1.8 and run on a machine using Windows 10, an i7 CPU at 2.1 GHz and 7GB of RAM dedicated to the JVM. The graph of the road network was created from OSM data. The only modifications made were that extra nodes were added to ensure that nodes were not separated by more than 20m, and insuring that all toll information was correctly captured. The cost of each toll was obtained from the MTA [17] and the value used was the undiscounted cost for a car.



(a) Quickest route

(b) Shortest route with no tolls

(c) Shortest route

(d) Shortest route with no highways

Fig. 1: Different routes for the same multi-trip depending upon the driving style

104 test trips were created, each of which contained between 0 and 10 waypoints, with an average of 4.413 waypoints per trip. The waypoints were randomly selected from the original graph data. Openrouteservice was chosen to create the test routes because of its ability to create routes with different combinations of route choice including avoiding tolls, avoiding highways or no restrictions along with minimizing the travel time or travel distance, and because the latitude, longitude and timestamp of points along the route could be exported via the api. The routes range in length from 2.8 to 172.9 miles with an average of 59.8 miles and had a duration ranging from 269 to 18505 seconds, with an average of 5984 seconds. These routes were then sampled so that the points occurred at a regular interval of 15 seconds so as to simulate a GPS trace. A byproduct of the sampling was that except for a few cases the actual waypoint would not appear on the trace. Figure 1 shows a map of New York with the different routes taken for a multi-trip beginning at the location marked A and ending at B travelling via the waypoints 1 to 6 for each of 4 different driving styles. Table 1 details the cost functions used to generate the test routes.

The parameters for Algorithms 1 and 2 were set as follows: the heading tolerance α was set to 5°, while the allowable tolerance ε was set to 5 seconds \ 2% where the driving style algorithm incorporates shortest route by time and set to 0.03 miles \ 2% where the driving style algorithm incorporates shortest route by distance. These values were chosen based on the rectangular street layout in the majority of New York and the approximate size of a city block. The cost functions χ were instantiated as follows: The penalties for passing through a toll booth when the driving style indicates that the driver avoids tolls is 10 minutes \ 3 miles for each dollar of the toll cost, and the penalty for traversing an edge tagged as highway is the cost of traversing the edge multiplied by 30 regardless of cost measure. It should be noted that these are not the same as the functions used by openroutesource. For example our cost functions allow, but penalise, the use of tolls and highways while openrouteservice avoid tolls and highways. The corresponding approximate functions are shown in Table 1.

openrouteservice Cost Function	How Calculated	Approximate Cost Function
Q	Quickest route	Q'
S	Shortest route	S'
QNH	Quickest route avoiding highways	QNH'
SNH	Shortest route avoiding highways	SNH'
QNT	Quickest route avoiding tolls	QNT'
SNT	Shortest Route avoiding tolls	SNT'

Table 1: Descriptions of cost functions

5 Results

True Driving Style			Driving Style Algorithm					
			Q'	S'	QNH'	SNH'	QNT'	SNT'
	Q	% Intervals	102.17	167.77	514.81	534.64	115.90	176.91
		Fewest Int.	95.2	8.7	3.8	2.9	61.5	7.7
		Precision	0.945	0.536	0.163	0.145	0.829	0.498
		Accuracy	0.960	0.753	0.576	0.561	0.907	0.732
		Recall	0.965	0.893	0.841	0.776	0.961	0.880
		Vagueness	129.32	218.86	91.81	88.72	128.35	202.35
	S	% Intervals	313.9	99.6	362.8	224.2	327.2	114.2
		Fewest Int.	4.8	95.2	3.8	12.5	4.8	61.5
		Precision	0.285	0.958	0.246	0.403	0.272	0.821
		Accuracy	0.638	0.961	0.620	0.696	0.632	0.893
		Recall	0.893	0.954	0.893	0.904	0.891	0.937
		Vagueness	166.86	161.37	119.43	118.83	166.19	142.99
	QNH	% Intervals	257.7	154.5	106.3	130.0	261.7	157.1
		Fewest Int.	7.7	12.5	85.6	30.8	6.7	11.5
		Precision	0.368	0.604	0.921	0.740	0.360	0.581
		Accuracy	0.687	0.795	0.947	0.856	0.682	0.779
		Recall	0.948	0.933	0.970	0.933	0.941	0.913
		Vagueness	258.93	235.71	85.68	194.71	249.44	210.45
	SNH	% Intervals	394.1	135.1	295.0	102.0	400.0	139.0
		Fewest Int.	2.9	26.0	5.8	97.1	26.9	2.9
		Precision	0.229	0.705	0.312	0.940	0.226	0.672
		Accuracy	0.614	0.847	0.655	0.955	0.612	0.857
		Recall	0.904	0.952	0.919	0.959	0.904	0.935
		Vagueness	179.20	235.89	126.81	128.99	177.57	207.93
	QNT	% Intervals	114.8	177.6	525.3	544.7	104.6	169.3
		Fewest Int.	70.2	8.7	3.8	2.9	98.1	9.6
		Precision	0.839	0.501	0.161	0.145	0.923	0.524
		Accuracy	0.912	0.736	0.575	0.562	0.950	0.745
		Recall	0.963	0.889	0.848	0.789	0.965	0.887
		Vagueness	176.01	232.77	91.89	87.50	119.57	208.45
	SNT	% Intervals	324.8	111.6	372.6	232.0	320.9	98.7
		Fewest Int.	4.8	62.5	3.8	11.5	4.8	98.1
		Precision	0.280	0.863	0.241	0.393	0.284	0.974
		Accuracy	0.638	0.923	0.618	0.693	0.641	0.970
		Recall	0.909	0.961	0.898	0.913	0.911	0.962
		Vagueness	173.34	227.76	120.45	275.42	164.07	123.27

Table 2: Efficiency of Driving Style Algorithms

First we measure the performance of each driving style cost function in estimating waypoints for each of the true driving styles by running Algorithm 1 on each combination of driving style algorithm and true driving styles. The results are shown in Table 2. From this table it can be seen that the driving style algorithm that most closely resembles the true driving style consistently produces the smallest number of intervals as a percentage of the number of actual waypoints and that it produced the fewest (or equal fewest) intervals in at least 85% of cases. The vagueness measure is defined as the average width of the intervals returned in seconds. We note that in some some cases a different driving style returns a lower value for vagueness but this is outweighed by the number of intervals returned. Other measures of performance are the precision, accuracy and recall of each driving style algorithm against the true driving style of the route. A trace contains n waypoints. A search returns m explicit intervals as predicted waypoints which gives $m + 1$ additional implicit intervals which predict no waypoint. We measure the number of waypoints we correctly estimate (tp), the number of waypoints we falsely estimate (fp) and the number of waypoints we miss (fn). The standard measure of accuracy does not work because the algorithm output is an arbitrary number of intervals, therefore we calculate accuracy as the number of explicit and implicit intervals which are correct (i.e. explicit intervals that contain a true waypoint, and implicit intervals that do not contain a true waypoint). The formula for precision, accuracy and recall are as follows.

$$Precision = \frac{tp}{m}$$

$$Accuracy = \frac{2m + 1 - fp - fn}{2m + 1}$$

$$Recall = \frac{tp}{n}$$

It can be seen that the precision, accuracy and recall are at their highest when the driving style algorithm most closely resembles the true driving style.

	True Driving Style					
	Q	S	QNH	SNH	QNT	SNT
% Intervals	102.0	99.6	105.4	102.0	104.4	98.7
Precision	0.940	0.958	0.901	0.940	0.914	0.971
Accuracy	0.950	0.956	0.927	0.950	0.935	0.965
Recall	0.959	0.954	0.950	0.959	0.954	0.959
Vagueness	134.36	162.61	108.25	131.85	129.21	129.93

Table 3: Efficiency of Waypoint Estimation Algorithm Assuming No Driving Style Information when we use the maximum width of intervals as tie-breaker

	True Driving Style					
	Q	S	QNH	SNH	QNT	SNT
% Intervals	102.0	99.6	105.4	102.0	104.4	98.7
Precision	0.938	0.950	0.909	0.936	0.921	0.971
Accuracy	0.948	0.948	0.935	0.946	0.942	0.965
Recall	0.957	0.946	0.959	0.954	0.961	0.959
Vagueness	127.02	149.77	84.55	125.19	120.53	121.75

Table 4: Efficiency of Waypoint Estimation Algorithm Assuming No Driving Style Information when we use the minimum width of intervals as tie-breaker

We now consider Algorithm 2 which considers all simulated driving styles and returns the predictions from the style that produced fewest intervals, with the results shown in Tables 3 and 4. Minimum spread shows an average reduction of 9% in vagueness for a loss of less than 1% in precision, accuracy and recall when compared to maximum spread. The average cover of minimum spread intervals represent less than 9% of the actual trace. Therefore minimum spread is the better option. Examining the results in Table 4 show that when we do not know the driving style, we can still achieve a precision greater than 93% on average and a recall greater than 95%. Thus our method can infer intermediate waypoints in a single multi-trace without knowledge of true driving style.

6 Conclusion

In this paper we have identified a method to infer intervals which contain waypoints from a multi-trip GPS trace without knowing the driving style of the person creating the trace, using only a graph of the area. We show the simulated driving style which is closest to the true driving style produces the highest precision, accuracy and recall for each of the true driving styles. Our algorithm is based on multiple searches, using one for each possible driving style. It then selects which to return, selecting the list with the fewest estimated intervals and breaking ties by selecting the list with the minimum spread. In empirical testing the algorithm had a recall of over 95% with a precision of over 93% on average, while the intervals on average cover less than 9% of the trace.

Future work will involve investigating an approach for inferring waypoints on a GPS trace where the driver does not follow a single driving style, but may use a combination of driving styles over the duration of their trip, and also to investigate the possibility of integrating the algorithm into data analytic methods for improving destination prediction.

Acknowledgement. This project has been funded by Science Foundation Ireland under Grant Number SFI/12/RC/2289.

References

1. D. A. Desmond, K. N. Brown.: Inferring Waypoints Using Shortest Paths. In: Proc. of the 24th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2016): 45-56 (2016)
2. J. G. Wardrop.: Some theoretical aspects of road traffic research. In: Proceedings of Institute of Civil Engineers, Part II: 325-378 (1952)
3. J. R. Duffel, A. Kalombatis.: Empirical studies of car driver route choice in Herfordshire. In: Traffic Engineering and Control 29 (7/8), 398-408 (1988)
4. R. Huchingson, R. McNees, C. Dudek.: Survey of motorist route-selection criteria. In: Transportation Research Record 643, 45-48 (1977)
5. Chen, T.Y., Chang, H.L. and Tzeng, G.H.: Using a weight-assessing model to identify route choice criteria and information effects. In: Transportation Research Part A: Policy and Practice, 35(3): 197-224. (2001)
6. S. Zhu, D. Levinson.: Do People Use the Shortest Path? An Empirical Test of Wardrop's First Principle. In: Proc. of the 91st Annual Meeting of the Transportation Research Board, Washington, DC. (2012)
7. Tanaka, K., Kishino, Y., Terada, T., Nishio, S.: A destination prediction method using driving contexts and trajectory for car navigation systems. In: Proceedings of ACM Symposium on Applied Computing (2009)
8. Alvarez-Garcia, J.A., Ortega, J.A., Gonzalez-Abril, L., Velasco, F.: Trip destination prediction based on past GPS log using a hidden markov model. In: Expert Systems with Applications: An International Journal, vol. 37 (2010)
9. Xue, A.Y., Zhang, R., Zheng, Y., Xie, X., Huang, J., Xu, Z.: Destination prediction by sub-trajectory synthesis and privacy protection against such prediction In: Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013) (2013)
10. Kafsi, M., Grossglauser, M., Thiran, P.: Traveling Salesman in Reverse: Conditional Markov Entropy for Trajectory Segmentation In: ICDM 2015: 201-210 (2015)
11. https://www.google.com/maps/
12. https://www.openstreetmap.org/
13. http://map.project-osrm.org/
14. https://graphhopper.com/maps/
15. https://go.openrouteservice.org/
16. Dijkstra, E.W.: A note on two problems in connexion with Graphs. In: Numerische Mathematic, 1:269-271 (1959)
17. http://web.mta.info/bandt/traffic/btmain.html/