Scalable Unsupervised Multi-Criteria Trajectory Segmentation and Driving Preference Mining

Barth, Florian; Funke, Stefan; Skovgaard Jepsen, Tobias; Proissl, Claudius

Published in:
BIGSPATIAL '20: Proceedings of the 9th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data

DOI (link to publication from Publisher):
10.1145/3423336.3429348

Publication date:
2020

Document Version
Early version, also known as pre-print

Scalable Unsupervised Multi-Criteria Trajectory Segmentation and Driving Preference Mining

Florian Barth
Universität Stuttgart, Germany

Stefan Funke
Universität Stuttgart, Germany

Tobias Skovgaard Jepsen
Aalborg University, Denmark

Claudius Proissl
Universität Stuttgart, Germany

ABSTRACT

We present analysis techniques for large trajectory data sets that aim to provide a semantic understanding of trajectories reaching beyond them being point sequences in time and space. The presented techniques use a driving preference model w.r.t. road segment traversal costs, e.g., travel time and distance, to analyze and explain trajectories.

In particular, we present trajectory mining techniques that can (a) find interesting points within a trajectory indicating, e.g., a via-point, and (b) recover the driving preferences of a driver based on their chosen trajectory. We evaluate our techniques on the tasks of via-point identification and personalized routing using a data set of more than 1 million vehicle trajectories collected throughout Denmark during a 3-year period. Our techniques can be implemented efficiently and are highly parallelizable, allowing them to scale to millions or billions of trajectories.

1 INTRODUCTION

The ubiquity of mobile devices with position tracking capabilities via GPS or localization using WiFi and mobile networks continuously generate vast streams of location data. Such data may be used in a variety of ways. Mobile networks providers and many companies, such as Google or Apple, use the location data of their customers to improve their services, e.g., by monitoring of traffic flow or detection of special events. Location data sharing platforms such as Strava, GPSies, and OpenStreetMap (OSM) allow their users to share their location data with their community. In all of these cases, location measurements are considered collectively as sequences, each reflecting the movement of a person or a vehicle. Such sequences can be map-matched to paths in an underlying transportation network—in our case a *road network*—using appropriate methods [21]. We refer to such map-matched sequences as *trajectories* throughout the paper.

A common assumption is that most of the time, users travel on 'optimal' routes towards a (possibly intermediate) destination, where optimality is understood as the shortest path w.r.t. suitable scalar *traversal costs* of each road segment in the underlying road network. For instance, route planners and navigation systems often use travel times as traversal costs. However, in practice, drivers seldom travel on such 'optimal' routes due to complex traversal costs, e.g., time-dependent and uncertain travel times [17], a (possibly unknown) combination of several traversal costs [7], or due to changing intentions/destinations during a trip. We therefore investigate analysis techniques that do not rely on a fixed criterion but are capable of identifying a suitable combination of given criteria.

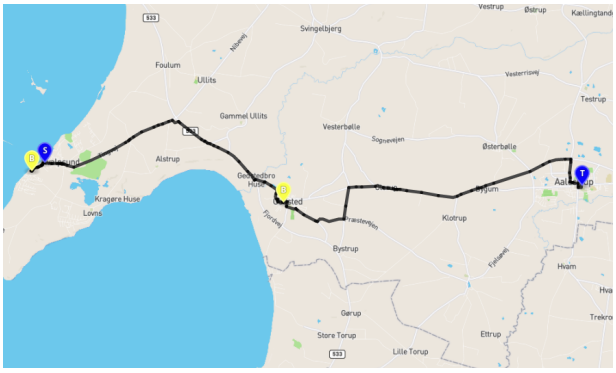


Figure 1: An example of a trajectory going from S to T with two intermediate stops that are labeled B.

The high-level goal of this paper is to develop trajectory mining techniques to enable a better understanding of the semantics of trajectory data. Concretely, we focus on the tasks of trajectory segmentation and driving preference mining.

Trajectory Segmentation. A trajectory is often not just the manifestation of someone going from location S to location T following an optimal route w.r.t. some criteria, but rather determined by a sequence of activities/intentions. For instance, Figure 1 shows a trajectory from S to T with two intermediate stops labeled B. The driver starts at location S but rather than taking the fastest routes, decides to drive southwest and makes a stop. Then, the driver backtracks and takes the fastest route from S to T but decides to make a stop on the way. In this paper, we present a trajectory segmentation approach that can identify intermediate stops or other points of interest in a trajectory and divide it into subtrajectories accordingly.

In contrast to previous work on trajectory segmentation [1, 4, 8, 15, 18], our approach solely relies on traversal costs and the structure of the road network. No additional information such as time stamps are required. Thus, compression techniques for efficient trajectory storage [16, 19] are applicable. However, despite not utilizing temporal information, our experiments show that our trajectory segmentation approach can recover such information through a structural analysis of the trajectory. In addition, our trajectory segmentation approach uses a driving preference model to divide trajectories into subtrajectories. To the best of our knowledge, this is the first trajectory segmentation approach to do so.

Driving Preference Mining. Given a trajectory, it is an interesting question which criteria the driver likely tried to optimize. We refer

to this combination as a driving preference. We show how a known algorithm for driving preference mining [10] can be made sufficiently robust to recover preferences from vehicle trajectories. We compare the modified algorithm with two benchmark functions and address the question, how well the recovered preferences describe the trajectories.

Related Work

In this paper, we consider two main applications: trajectory segmentation and driving preference mining.

Previous work on trajectory segmentation can broadly be categorized as supervised [1, 4] and unsupervised [8, 15, 18]. Supervised approaches require predetermined criteria. In contrast, unsupervised approaches find the combination of available criteria that best explain the driving behavior in the input trajectory. Our proposed trajectory segmentation approach is unsupervised. Thus, the required prior assumptions on drivers' behavior are reduced to a minimum. In addition, our approach requires no spatio-temporal information about the trajectories, unlike existing trajectory segmentation approaches [1, 4, 8, 15, 18].

Driving preference mining—also referred to as driving preference learning —has been studied previously as well.

A popular approach of modeling driving preferences is to consider them as random variables. For instance, preference mining methods based on Gaussian Mixture Models are presented in [20] and in [6]. Balteanu et al. [3] present a probabilistic method that compares the input trajectory with pareto-optimal trajectory sets. Campigotto et al. [5] show how mining driving preferences can be accomplished using Bayesian learning strategies.

Our approach is non-probabilistic and considers the driving preference as the solution of an optimization problem. Delling et al. [7] present a similar technique, where the driving preference is chosen such that the overlap of the given and the computed trajectory is maximal. The main difference to our approach is that Delling et al. focus on geographical similarities of trajectories while we define similarity in terms of their traversal costs.

The work of Funke et al. [10] is the most closely related work both w.r.t. to trajectory segmentation and driving preference mining. They present a method which decides whether there exists a conic combination of the traversal costs such that a given trajectory is optimal for this weighting—or *preference*—of the traversal costs, and outputs the preference. Their method can recover the driving preferences of synthetic trajectories where such a conic combination is guaranteed to exist. However, this is not guaranteed for real world trajectories due to changes in driver preferences within the trajectory or inaccuracy in the traversal costs. We extend their method to obtain robust preferences in case a trajectory is not optimal for any preference. In addition, we use their method in our trajectory segmentation approach to determine the start and end of a subtrajectory.

1.1 Our Contribution

We develop techniques to 'explain' route choices made by drivers based on several natural criteria and evaluate them on a large data set of real-world trajectories. In particular, we propose a simple method for unsupervised trajectory segmentation that is able to approximate locations where drivers change their intentions/destinations along their trajectories. In contrast to previous work, our approach does not require spatio-temporal information about the trajectories, which considerably reduces the storage requirements.

Additionally, we present a simple, yet effective modification of a driving preference mining technique that allows to estimate drivers' preference that is more robust with respect to noise or sporadic 'suboptimal' routing decisions.

Both our approaches are built on the algorithm presented in [10] which deduce driving preferences from trajectories. As a result, driving preferences are integral to both our trajectory segmentation and driving preference mining techniques, demonstrating the close relationship between these two applications. To the best of our knowledge, we are the first to show this relationship.

2 PRELIMINARIES

2.1 Data Set

2.1.1 Road Network Data. We use a directed graph representation of the Danish road network [13] $G = (V, E)$ that has been derived from data provided by the Danish Business Authority and the OSM project. In this graph representation, V is a set of nodes, each of which represents an intersection or the end of a road, and E is a set of edges, each of which represents a directed road segment. The graph representation of the Danish road network contains the most important roads and has a total of 583,816 intersections and 1,291,171 road segments. In addition, each road segment has attributes describing their length and type (e.g., motorway) and each intersection has attributes that indicate whether they are in a city area, a rural area, or a summer cottage area. The data is further augmented with a total of 163,044 speed limits combined from OSM data and speed limits provided by Aalborg Municipality and Copenhagen Municipality [14].

2.1.2 Trajectory Data. We use a set of 1,306,392 vehicle trajectories from Denmark collected between January 1st 2012 and December 31st 2014 [2]. The trajectories have been map-matched to the graph representation of the Danish road network s.t. each trajectory is a sequence of traversed road segments $T = (e_1, \ldots, e_n)$ where $e_i \in E$ for $1 \leq i \leq n$. In addition, each segment is associated with a time stamp and a recorded driving speed whenever the GPS data is sufficiently accurate. In this data set, a trajectory ends after its GPS position has not changed more than 20 meters within three minutes. See [2] for more details.

Trajectory Stitching. A vehicle trajectory in the trajectory data set ends when the vehicle has not moved more than 20 meters within three minutes. However, in practice, a driver may choose a trajectory with several intermediate stops, for instance when visiting multiple supermarkets to go grocery shopping. We are interested in examining such trajectories. We therefore stitch temporally consecutive trajectories from the same vehicle together if there is less than 30 minutes difference between the end of the current trajectory to the start of the next. Each stitch thus indicates the end of a 3 to 33 minutes break in movement. We call these stitches *break points* that mark a temporal gap in the trajectory.

In many cases temporally consecutive trajectories are not connected due to imprecision or lack of GPS data. In such cases, we

compute the shortest route from the destination of the current trajectory to the start of the next. If the shortest route is shorter than 200 meters or consists of at most one road segment, we stitch the trajectories. We continue attempting to join the stitched to the next trajectory until the next trajectory does not meet the stitching criteria. See Appendix A for further details.

From the original 1,306,392 trajectories we obtain 260,190 combined trajectories. Of these trajectories, 190,199 trajectories are stitched and contain break points.

2.2 Routing Cost Types

From the data sets described in Section 2.1, we derive a number of criteria that are a measure of the expected cost of taking a route. In our experiments, we use the following four cost types: travel time, congestion, crowdedness, and number of intersections. We normalize the average value of each cost type to one.

Travel Time. Each road segment is associated with a fixed value that represents the time it takes to traverse the road segment. To derive travel time, we combine historical traversal data from the trajectory data set with travel time estimates from a pre-trained machine learning model [13]. See Appendix B.1 for further details.

Congestion. We derive the congestion level on a particular road segment based on how close to the speed limit people tend to drive. The closer to the speed limit, the less congestion. Many road segments do not have a speed limit in our speed limit data set. In such cases, we use a simple OSM routing heuristic, see Appendix B.2.

Crowdedness. This criterion measures how 'crowded' the surroundings along a vehicle trajectory are. We derive a crowdedness value for each road segments from the number of nearby road segments and points of interest OSM nodes. Further details can be found in Appendix B.3.

Number of Intersections. The number of intersections visited in a trajectory, excluding the source intersection.

2.3 Personalized Routing

The notion of shortest path requires some distance or cost measure to compare paths (i.e., routes) in a road network. To this end, we use the notion of personalized routing from [12] which takes into account both multiple traversal cost types and driving preferences.

2.3.1 Personalized Cost. The personalized cost of a road segment e combines a d-dimensional *cost vector* $c(e) = (c_1(e), \ldots, c_d(e))$ and a d-dimensional *preference vector* $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ where $\alpha_i \geq 0$ and $\sum \alpha_i = 1$. Each cost $c_i(e)$ for $1 \leq i \leq d$ is a measure of the cost to traverse road segment e and each preference α_i represent the drivers preference w.r.t. to minimizing cost $c_i(e)$. For instance, $c_1(e)$ and $c_2(e)$ may be the travel time and the number of intersections of the road segment e. Assuming equal scale of the costs, a preference vector of $(0.7, 0.3)$ indicates that a driver values travel time more than the number of intersections.

The personalized cost of a route $\pi = (e_1, e_2, \ldots, e_k)$ in a road network is $p(\pi \mid \alpha) = \sum_{i=1}^{k} p(e_i \mid \alpha)$ where $p(e_i \mid \alpha)$ is the personalized cost of road segment e_i and α represent the driving preferences of a particular driver. Given driving preferences α, the personalized cost of a road segment e is $p(e \mid \alpha) = \sum_{i=1}^{d} \alpha_i c(e)$. We call a path π from s to t *personalized path* if $\forall \pi' \in \Pi : p(\pi \mid \alpha) \leq p(\pi' \mid \alpha)$, where Π is the set of all paths from s to t.

2.3.2 Deducing Routing Preferences. Given a trajectory T in the personalized route setting a natural question to ask is if it is a personalized path for some preference α. The trajectory $T = v_0 v_1 \ldots v_{k-1}$ is a personalized path if a solution exists to the following LP with variables $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ representing driving preferences [10].

$$\begin{aligned} &\text{Minimize} && 1 \\ &\text{subject to} && \forall \pi \in \Pi \colon p(T \mid \alpha) - p(\pi \mid \alpha) \leq 0 \\ & && \sum_{i=1}^{d} \alpha_i = 1 \\ & && \forall i \in \{1, \ldots, d\} \colon \alpha_i \geq 0 \end{aligned} \tag{1}$$

The LP contains a constraint for *each* possible route $\pi \in \Pi$ from v_0 to v_{k-1}. These constraints require the personalized costs of T for the solution, the preference vector α, to be lower than those of every other path π. Note that no objective function is used and T is therefore a personalized path if any preference vector α exists that satisfies all the constraints.

The Dijkstra Oracle. The solution to the LP in Equation (1) is a preference vector α for which no path from v_0 to v_{k-1} has lower personalized cost than the trajectory T. Writing down the complete LP for *all possible* paths from v_0 to v_{k-1} is infeasible. Fortunately, it suffices to add the constraints one by one via a so-called separation oracle [10].

In brief, the LP is first solved using only the last two constraints in Equation (1). This results in some initial preference vector α. Then, α is verified to satisfy all the constraints in Equation (1) by simply running a Dijkstra from v_0 to v_{k-1} with preference vector α. If so, a solution to the linear program in Equation (1) has been found and no further processing is required. If not, a violating constraint is discovered and added to the LP.

While this method finds a preference method α for T, if any exists, real world trajectories are often not personalized paths.

2.4 Trajectory Segmentation

In this section, we discuss the definition of the trajectory segmentation problem and a general algorithmic framework for it.

2.4.1 Trajectory Segmentation Problem. The segmentation of a trajectory $T = v_0 v_1 \ldots v_{k-1}$ is a sequence of B trajectory segments $S_1 = v_0 v_1 \ldots v_{b_1}$, $S_2 = v_{b_1} v_{b_1+1} \ldots v_{b_2}$, $S_3 = v_{b_2} v_{b_2+1} \ldots v_{b_3}$, up to $S_B = v_{b_{B-1}} v_{b_{B-1}+1} \ldots v_{b_B}$. We refer to the common node of two consecutive trajectory segments, e.g., S_1 and S_2, as a *segmentation point*. For instance, v_{b_1} is a segmentation point because it is at the end of S_1 and the start of S_2. Buchin et al. [4] define the segmentation problem as finding a (minimal) number of segments for a trajectory such that each segment fulfills a criterion. They provide a general algorithmic framework for arbitrary segmentation criteria.

2.4.2 Trajectory Segmentation Framework. In the framework of Buchin et al. [4], one has to provide a test procedure which verifies if a given segment meets the desired criterion. This test procedure is then used to repeatedly, greedily find the longest prefix that meets the criterion. The authors prove that this approach leads

to an optimal, i.e., minimal, segmentation for *monotone* criteria in $O(T(k)\log k)$ time if the test procedure takes $T(m)$ time for a segment of length m. They define monotonicity for a criterion as follows. If any segment $S \subseteq T$ satisfies the criterion, then any segment $S' \subseteq S$ also satisfies the criterion. Even though Buchin et al. [4] focus on self-similarity criteria for the segments, this definition adapts well to the optimality criterion we introduce in Section 3.1.

3 MULTI-CRITERIA TRAJECTORY SEGMENTATION

A driver may have several via-points on a trip. Sometimes, these via-points are linked to a point-of-interest such as gas station, but that is not necessarily the case. The trajectory segmentation approach we present in this section is designed to find all interesting via-points along a trajectory.

In brief, our approach assumes that drivers choose personalized paths between their via-points. Any deviation from their personalized path along a trajectory that goes from s to t indicates some interesting point p in the trajectory. The point of deviation p is marked as the end of the first trajectory segment and the beginning of the next. This process is repeated on the remaining subtrajectory going from p to t and so forth.

3.1 The Personalized Path Criterion

For trajectory segmentation in the framework of [4], described in Section 2.4.2, we propose a type of criteria that uses only the underlying graph and does not need any predetermined parameters. The *optimal path criterion* requires each trajectory segment S of a trajectory T to be an optimal path according to the traversal costs in the underlying graph. This criterion is monotone as defined in Section 2.4 because it requires S to be a "shortest path" and subpaths of shortest paths are also shortest paths. As a test procedure for a segment, we use a Dijkstra query.

The optimal path criterion can be generalized to the *personalized path criterion*. The personalized path criterion requires each trajectory segment to be a personalized path with respect to some driver preference α. This criterion is satisfied if there exists a solution to the LP in Equation (1). Note, that the α for each trajectory segment can differ.

Fixing Edge Cases. It is possible that there exists a minimal trajectory segment $S \subseteq T$ (consisting of a single road segment) which is not a personalized/optimal path. One road segment (u, v) might be more expensive in every traversal cost type than another path from u to v. This indicates that the used traversal cost types can not explain driver behavior for taking such an road segment. For the personalized path criterion, this can be remedied by including a cost type for which each road segment is a personalized path between its source and target intersections. In general, a unit cost type (every road segment e has $c_i(e) = 1$) has this property. This guarantees segmentability for arbitrary trajectories and makes the personalized path more robust. In our experiments, the number of intersections cost type fulfills this role.

This does, however, not fix the special case of self-loop edges, which in our data set typically represent road segments that allow traversals in parking lots. Such road segments can never be optimal because the optimal path from the source intersection to itself remains at the intersection. Self-loop edges can either be dealt with by deleting them from the trajectories, if they do not cover significant areas in the road networks, or by representing such road segments as two edges that each represent partial traversal of the self-looping road segment.

3.2 Experiments

We now investigate the capabilities of the trajectory segmentation method to identify via-points in trajectories on the basis of the trajectory data set described in Section 2.1. In particular, we use the stitched trajectory set to evaluate our trajectory segmentation approach, Personalized Path Trajectory Segmentation (PPTS), to the Optimal Path Trajectory Segmentation (OPTS) baseline which uses only a single cost type to check for the optimal path criterion. We consider the four variants OPTS-TT, OPTS-Con, OPTS-Int, and OPTS-Cro, that use the travel time, congestion level, number of intersections, and crowdedness, respectively, as the single cost type.

We compare PPTS's ability to segment trajectories to that of the baselines. Our comparison is both in terms of the number of trajectories that can be segmented and the ability of the trajectory segmentation algorithms to recover the break points in the stitched trajectory set. As mentioned in Section 2.1.2, these break points indicate a break of 3 to 33 minutes and are therefore likely to indicate a via-point within the trajectory. We discard the self-loop edges within each trajectory to increase segmentability, as described in Section 3.1. Typically, these self-loop edges represent road segments that allow driving around parking lots.

All algorithms used in our experiments are implemented in the Rust programming language[1]. We make the implementation of our method, the used graph and some example trajectories publicly available[2]. We use contraction hierarchies (CH) [11] to speed up the Dijkstra queries by orders of magnitude.

3.2.1 *Evaluation Functions.*

We use several evaluation functions to evaluate our trajectory segmentation method and for comparison with the baselines.

Segmentability Score. The segmentability score, or simply S-score, measures the proportion of trajectories that are segmentable by a trajectory segmentation algorithm. Ideally, the S-score is 100% indicating that all trajectories in the data set could be segmented by the used trajectory segmentation algorithm.

Break Recovery Rate. The Break Recovery Rate (BRR) is a measure of how good a trajectory segmentation algorithm is at placing segmentation points s.t. they coincide with known break points in the stitched trajectories. Let BP denote the set of known break points in a trajectory T and let SP denote the set of segmentation points output by a trajectory segmentation algorithm that has been given trajectory T as input. Then, the BRR of trajectory T is

$$\mathrm{BRR}(T) = \frac{|RBP|}{|BP|}$$

where $RBP = BP \cap SP$ is the set of recovered break points.

[1] https://www.rust-lang.org/
[2] https://github.com/Lesstat/ppts

Table 1: Mean algorithm performance on all stitched trajectories (ALL) and the 60,249 commonly segmentable trajectories (CS) that can be segmented by all algorithms.

	ALL		*CS*		
Algorithm	*BRR*	*S-score*	*BRR*	*SR*	*SQ-score*
PPTS	57.98%	**100.0%**	56.29%	**2.39**	**0.235**
OPTS-TT	34.62%	58.16%	**58.35%**	2.83	0.206
OPTS-Con	29.32%	49.61%	57.49%	3.99	0.144
OPTS-Int	**59.19%**	**100.0%**	57.61%	4.37	0.132
OPTS-Cro	35.86%	61.11%	58.10%	5.62	0.103

Segmentation Rate. A trajectory segmentation algorithm can achieve a high BRR by simply segmenting a trajectory into trajectory segments consisting of one road segment each. Although such a segmentation is guaranteed to recover all break points, it is also very likely to contain a lot of noise in the form of many false positives or false break points. To measure such noise, we use the Segmentation Rate (SR) which measures the number of segmentation points per break point:

$$\mathrm{SR}(T) = \frac{|SP|}{|BP|}$$

Ideally, the SR should be 1 for a trajectory segmentation that recovers all break points, i.e., has a BRR of 100%.

Segmentation Quality Score. The segmentation quality score, or simply SQ-score, is a summary score to measure the overall quality of a trajectory segmentation. It combines the BRR and SR as follows:

$$\mathrm{SQ}(T) = \frac{\mathrm{BRR}(T)}{\mathrm{SR}(T)}$$

Note, that the unit of the SQ-score is recovered break points per segmentation point and should ideally be 1.

3.2.2 Results. The results of our experiments are shown in Table 1.

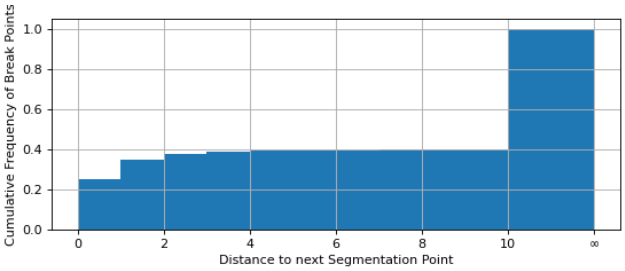
Segmentability. As shown in Table 1, PPTS and OPTS-Int are both capable of segmenting all trajectories and achieve an S-score of 100%. This result is not too surprising, since both algorithms use a unit cost type—the number of intersections—which guarantees that any trajectory can be segmented by these approaches, as discussed in Section 3.1. The remaining algorithms cannot segment a large portion of the trajectories (more than half in the case of OPTS-Con) and therefore achieve comparatively low S-scores. Thus, the inclusion of additional cost types can increase segmentability.

Segmentation Quality. As shown in Table 1, PPTS and OPTS-Int achieve similar BRRs that are substantially higher than the remaining OPTS variants. However, for a fair comparison that ignores the ability of the algorithms to segment trajectories, we have computed BRRs, SRs, and SQ-scores on the subset of trajectories that are commonly segmentable, i.e., the trajectories that can be segmented by all algorithms. On this subset, the BRRs of all algorithms are comparable. This suggests that the superior BRR when considering all trajectories for PPTS and OPTS-Int can largely be attributed to their greater capability for segmenting trajectories.

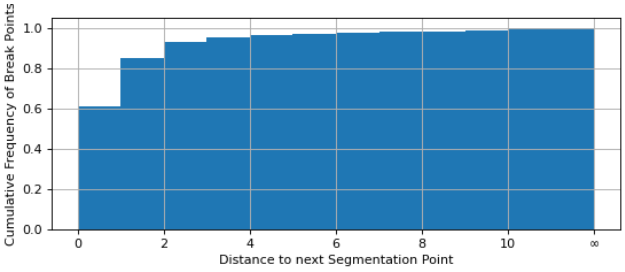
Although the BRRs are quite similar on the commonly segmentable trajectories, the SRs are quite different, as shown in Table 1 In particular, the OPTS-Con, OPTS-Int, and OPTS-Cro algorithms have, respectively, 67%, 83%, and 135% more segmentation points per break point than PPTS. The SR of OPTS-TT algorithm is just 18% higher than that of PPTS.

The higher SRs of the single-cost-type baselines compared to PPTS suggest that the inclusion of driving preferences and multiple criteria reduces the amount of false positives. The PPTS achieves the lowest BRR on the commonly segmentable trajectories, but, as shown in Table 1, PPTS achieves the best overall trajectory segmentation quality with an SQ-score of 0.235, since it introduces the fewest false break points and thus have the lowest SR. Conversely, OPTS-TT achieves the highest BRR score on the same data subset, but introduces more false break points than PPTS. As a result, OPTS-TT achieves only the second-highest segmentation quality with an SQ-score of 0.206. Still, our results support the wide-spread use of the travel time cost type in many routing services, but also show that taking additional cost types and driving preferences into account can lead to better trajectory segmentation.

For the sake of brevity, we present only the comparison between PPTS and the best-performing baseline, OPTS-TT, in the remainder of this section.



(a) OPTS-TT



(b) PPTS

Figure 2: Distribution of distance between a break point and the next segmentation point for (a) OPTS-TT and (b) PPTS. Break points in trajectories without any segmentation point are assigned distance ∞.

Segmentation Point Accuracy. We have thus far only considered exact break point recovery, but a segmentation may still be useful if it indicates that a break point is near. Figure 2 shows the percentage of break points which is within a certain (hop) distance to the next

segmentation point for OPTS-TT and PPTS. PPTS places segmentation points considerably more accurate than OPTS-TT. PPTS places more than 60% of the break points within one road segment of the nearest segmentation point and over 80% are within two road segments of the next segmentation point. OPTS-TT achieves less than half of PPTS's performance. However, the performance disparity illustrated in Figure 2 is largely due to better segmentability of trajectories when using PPTS. If the analysis is restricted to break points with distance $d < \infty$ to the nearest segmentation point, i.e., trajectories that are segmentable by OPTS-TT, their distributions are comparable.

Qualitative Segmentation Assessment. A good trajectory segmentation marks break points (or other interesting points) along a trajectory with a segmentation point. However, a good trajectory segmentation should also avoid too many false positives.

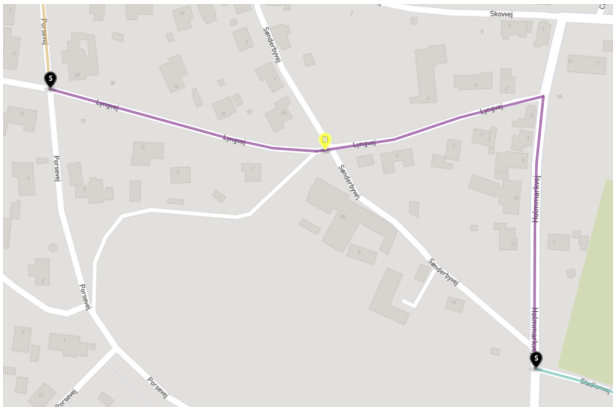
The PPTS and OPTS-TT, respectively, have an SR of 2.39 and 2.83 segmentation points per break point, respectively. However, although these numbers suggest that there are more false break points when using OPTS-TT, our data only contains positive examples of interesting behavior within the trajectory, i.e., the break points in the stitched trajectories. As result, we cannot quantitatively determine whether the segmentation points that do not match a break point are indeed false positives or mark interesting, but unknown, behavior during the trajectory. We therefore qualitatively assess the validity of the segmentation of a few trajectories.

Figure 3 shows a break point (marked with 'B' in yellow) in a segmented trajectory. The segmentation by OPTS-TT shown in Figure 3a places two segmentation points (marked with 'S' in black) around the break point. These segmentation points fail to recover the break point but are both within a distance of two road segments of the break point. Thus, the OPTS-TT segmentation appear to detect the presence of the break point, but fails to place the segmentation points exactly. The PPTS segmentation shown in Figure 3b, is a better segmentation and recovers the break point exactly (indicated by the black marker labeled 'B').

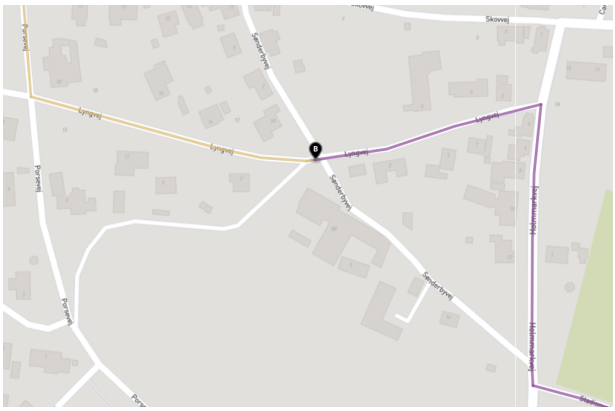
Figure 4 shows another part of the trajectory shown in Figure 3. Here, OPTS-TT places a segmentation point without comparable segmentation points in the PPTS segmentation. This additional segmentation point has no apparent meaning, and, upon detailed inspection, appears to occur due to inaccuracies in the estimated travel time in the area. This suggests that PPTS may be more robust than OPTS-TT to noise in the traversal cost data.

For the purposes of quantitative evaluation, our method attempts to recover breaks of 3 to 33 minutes from trajectories. However, our trajectory segmentation approach can discover interesting behavior beyond these known breaks. For instance, Figure 5 shows a segmentation point marking a detour to a gas station. This segmentation point is placed by both OPTS-TT and PPTS.

3.2.3 Processing Time. While using personalized routing does improve break recovery, it comes with an increase in processing time of trajectory segmentation. The increase in processing time for the personalized path variant is mostly driven by the CH-Dijkstra queries being slower.



(a) OPTS-TT



(b) PPTS

Figure 3: A break point in a trajectory and the segmentation points for (a) OPTS-TT and (b) PPTS. Yellow markers labeled 'B' indicate a break point and black markers labeled 'S' a segmentation point. A black marker labeled B indicates a break point that is recovered by a segmentation point.

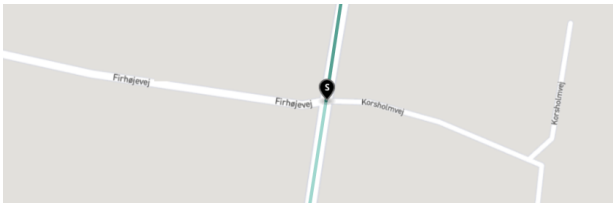


Figure 4: A segmentation point with no obvious event occurring.

The trajectory segmentation process is trivially parallelizable, since each trajectory can be processed independently, making segmentation of even billions of trajectories feasible. In our experiments, we parallelized the trajectory segmentation process across 64 cores, each with a clock speed of 2.3 GHz. The time to process the 190,199 stitched trajectories for single-criteria and multi-criteria trajectory segmentation is, respectively, 1 and 5 hours in total, and 19

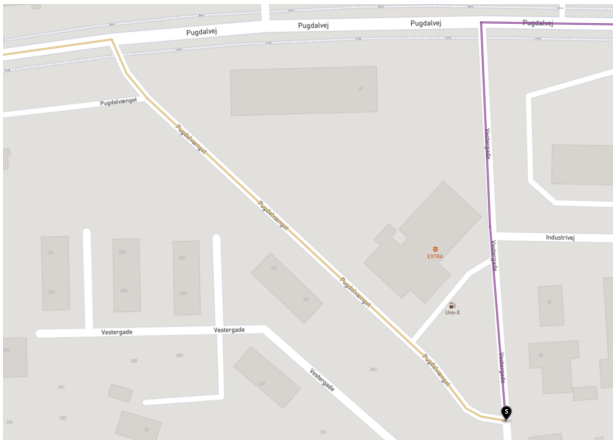


Figure 5: A segmentation point recovers a detour to a gas station that is not marked as a break in our data set.

and 95 milliseconds per trajectory on average. The total processing time took about half an hour in wall-clock time.

3.3 Discussion

Overall, PPTS achieves the highest trajectory segmentation quality in our experiments, followed by OPTS-TT. The results suggest that PPTS has two primary advantages over the baselines in our experiments. The use of multiple cost types and driving preferences makes PPTS capable of (1) segmenting more trajectories and (2) explaining driving behavior better, resulting in fewer false break points being placed. Our qualitative assessment of the OPTS-TT and PPTS segmentations support the conclusion that the segmentation points placed by PPTS are less likely to be false positives. In addition, PPTS discovered a detour to a gas station that is not indicated by a break point in our trajectory data set.

Even in the case where a break point is not recovered, PPTS is likely to place a segmentation point near the break point. PPTS places a segmentation point within a distance of 3 road segments for 95% of break points, but OPTS-TT for less than 40% of the break points. Although, this difference is largely explained by PPTS being capable of segmenting more trajectories, it suggests that PPTS's segmentation points are likely to indicate some interesting part of a trajectory. Although the increase in performance of PPTS over OPTS-TT comes at a factor 5 increase in processing time, it is still capable of segmenting a trajectory in a fraction of a second on average.

Stitching Parameters. Changing the parameters for the stitching process has only very little effect on our results and do not affect our conclusions. The results are virtually invariant to changes to the temporal stitching threshold. However, they are sensitive to changes to the stitch length threshold, although the effect is minor. The longer the stitches are allowed to be, the worse break recovery performance for both OPTS-TT and PPTS. This is likely because more noise is introduced when longer stitches are allowed.

4 ROBUST DRIVING PREFERENCE MINING

The trajectory segmentation approach presented in Section 3 implicitly recovers driving preferences α for each trajectory segment when determining the personalized path segments by searching for a solution to Equation (1). Although it is possible to collect the preference vectors for each trajectory segment, one is typically more interested in a *single* preference vector to describe a driver's general behavior for use in, e.g., personalized route planning [3, 12, 20]. Fortunately, this is possible with only a minor modification of the linear program in Equation (1):

$$\begin{aligned} &\text{Minimize} && \delta \\ &\text{subject to} && \forall \pi \in \Pi\colon p(T \mid \alpha) - p(\pi \mid a) \leq \delta \\ & && \sum_{i=1}^{d} \alpha_i = 1 \\ & && \forall i \in \{1, \ldots, d\}\colon \alpha_i \geq 0 \\ & && \delta \geq 0 \end{aligned} \tag{2}$$

As with Equation (1), this linear program may also be solved in polynomial time using the LP path oracle described in Section 2.3.2.

The following modifications have been made to Equation (2). First, the solution to the linear program is a preference vector α for the whole trajectory T rather than a trajectory segment. Second, by introducing δ to the first constraint, T is not required to be a personalized path w.r.t. to the α. Third, Equation (1) minimizes δ s.t. T is as close to being shortest-path optimal as possible w.r.t. to the preference vector α that is the solution to the linear program.

The effect of the modifications made in Equation (2) is that the linear program always has a feasible solution and therefore our approach always outputs some preference vector α. If $\delta = 0$, then the recovered α fully explains the driver behavior in the trajectory and is identical to the solution of Equation (1). Otherwise, if $\delta > 0$, then the recovered α does not fully explain the driver behavior but explains it as much as possible given the available traversal costs.

4.1 Experiments

We now evaluate our driving preference mining approach on a personalized routing task using the data set described in Section 2.1. Specifically, we evaluate our approach for each trajectory $T = v_0 v_1 \ldots v_{k-1}$ in a trajectory set as follows. First, we solve Equation (2) for T. Then, we use the resulting preference vector α to compute a personalized route (or personalized path) π from v_0 to v_{k-1}. Ideally, the preference vector α combined with the source v_0 and target v_{k-1} is sufficient to reconstruct or recover the route driven in trajectory T. We therefore refer to π as the *recovered route* of trajectory T.

W.r.t. the task of personalized routing, we are interested in measuring two qualities about our approach. First, how well do the recovered preference vectors model driving behavior. Second, how well do the preference vectors match the preferences of the drivers.

4.1.1 Evaluation Functions. To measure our approach's ability to model driving behavior, we use the Relative Recovered Route Overlap (RRRO):

$$\text{RRRO}(T, \pi) = \frac{|T \cap \pi|}{|T|}$$

Table 2: Mean RRRO and RCRS of RDP, TTP, and BRP of different algorithms for personalised routing on the unstitched trajectory set.

	RDP	*TTP*	*BRP*
RRRO	**0.74**	0.70	0.66
RCRS	**0.87**	0.85	0.81

Let α be a preference vector recovered from trajectory T. Here, π is the recovered route of trajectory T. If the preference vector α used to construct π fully captures the driving preferences exhibited in T, then the route π recovered using α should be identical to T, resulting in a relative recovered route overlap of 1.

To measure whether the preference vectors found using our approach match (actual) driver preferences, we use the Relative Cost Recovery Score (RCRS). The RCRS reflects the view that two routes are equivalent if their personalized costs are identical:

$$\text{RCRS}(T, \pi) = \frac{p(\pi \mid \alpha)}{p(T \mid \alpha)}$$

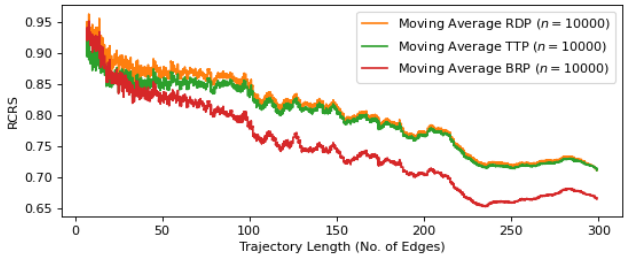
Note, that $p(T \mid \alpha) \geq p(\pi \mid \alpha)$ since π is the shortest path w.r.t. the personalized costs of the road segments for the given preference vector α. Thus, the RCRS is always between 0 and 1. An RCRS value of, e.g., 0.8, indicates that the preference vector α accounts for 80% of the personalized cost of trajectory T. If the preference vector α fully captures the driver's preferences, then the RCRS is 1.

4.1.2 Baselines. We refer to our approach as Recovered Driving Preference (RDP) and compare its performance with two baselines. The first baseline, Travel Time Preference (TTP), always returns a preference vector that has weight one for travel time. The second baseline, Best Random Preference (BRP), generates five random preference vectors for a trajectory, evaluates them and returns the preference with the best result. The BRP baseline is run independently for the two evaluation functions used in our experiments.

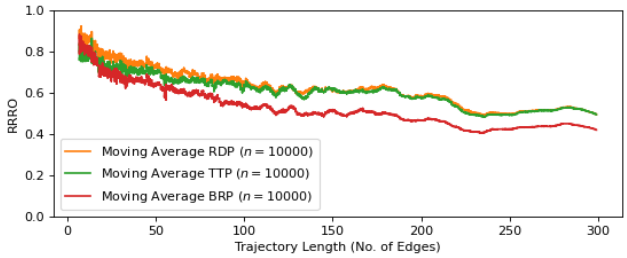
4.1.3 Results. We run our experiment on both the unstitched and stitched trajectory sets. The results on both trajectory sets are similar. For brevity, we report only the results on the unstitched trajectory set.

We summarize the results on the unstitched trajectories in Table 2. As shown in the table, RDP achieves both the highest mean RRRO and mean RCRS that are, respectively, 5.7% and 2.4% better than the best performing baseline TTP. We expect that these figures will be even higher if more than four traversal cost types are used.

The RDP shows superior performance compared to both baseline algorithms, but the performance of both RDP and the baselines deteriorate as trajectory length increases as shown in Figure 6. This is not particularly surprising since longer trajectories are more likely to contain more via-points, and none of the approaches in this experiment take such information into account. In addition, TTP approaches the performance of RDP as trajectory length increases. This suggests that people are likely to prioritize travel time more on long trips which matches both our expectations and anecdotal experiences.



(a) RCRS



(b) RRRO

Figure 6: The (a) RCRS and (b) RRRO scores of RDP, the TTP and the BRP with unstitched trajectories.

4.1.4 Robustness. We call our algorithm robust because, in contrast to the original algorithm given in [10], it does not require the trajectory to be a personalized path. Therefore, we are able to recover a driving preference for each of the 1,306,392 trajectories. On the contrary, the baseline algorithm is only able to process 590,251 trajectories, which is less than half. Hence, our modification is indeed a considerable robustness improvement.

4.1.5 Processing Time. We computed the results with a single core with a clock speed of 2.3 GHz. The average processing time is 1.04 milliseconds per trajectory and 0.02 milliseconds per road segment, but is proportional to the number of road segments in the trajectory.

4.2 Discussion

Our experiments show that the RDP preference vectors explain driver behavior better than the TTP and BRP baselines. Although the average results of TTP are very similar to those of RDP, they are never better. This is a strong indication that our approach indeed finds the best preferences to describe the drivers' behavior. In addition, the fast processing time (and trivial parallelizability) of RDP makes it scalable to even very large trajectory data sets.

Notably, the performance gap between RDP and TTP nearly disappears for long trajectories. This matches our expectation that travel time is the most important criterion for such trajectories.

5 CONCLUSION

In this paper, we have presented two techniques for large scale trajectory segmentation and driver preference mining. We have shown experimentally that our proposed trajectory segmentation

approach is a useful tool for understanding the semantics of a trajectory, e.g., the driver's intentions or changing destinations. In addition, our experiments showed that our proposed driver preference mining technique can indeed discover driver preferences for real trajectories and is sufficiently robust to process such data. Our techniques can be implemented efficiently in practice and are trivially parallelizable. Thus, they scale to very large trajectory sets consisting of millions or even billions of trajectories.

Interestingly, our approaches for trajectory segmentation and driving preference mining rely on the same model of driving preferences, showing that these two tasks are closely linked. To the best of our knowledge, we are the first to show this link.

Future Directions. There are many interesting directions for both our trajectory segmentation and driver preference mining approaches.

Our trajectory segmentation approach relies on linear combinations of costs. However, relationships between costs may be more complex and present an interesting opportunity for future work. In addition, driver preferences can be highly context-dependent and depend on, e.g., the time of day [20]. Extending our approach to utilize such contextual information is an important future direction.

In our driver preference mining approach, the linear program may have multiple solutions corresponding to a large set of preferences or a preference space. This makes it difficult to compare recovered preferences among trajectories or trajectory segments of the same trajectory. In particular, solving the linear program for two trajectories generated by two drivers with the same preferences may yield two different solutions, even if they also follow the same route. An important future direction is therefore to extend our driving preference mining technique to output identical (or at least similar) preferences in such situations. This will enable analysis of driver behavior through, e.g., driver preference clustering.

Finally, our driving preference mining approach assumes that the intent of the driver is to go straight from the start of the trajectory to the end of the trajectory. Thus, it ignores that such trajectories may have via-points. However, as demonstrated by our experiments, our trajectory segmentation approach can discover such via-points in trajectories. In future work, it would be interesting to explore synergies between our trajectory segmentation and our driving preference mining approach.

ACKNOWLEDGMENTS

This work was supported in part by the DiCyPS project, by grants from the Obel Family Foundation and VILLUM FONDEN, and in part by the Deutsche Forschungsgemeinschaft (DFG) within the priority program 1894: Volunteered Geographic Information.

REFERENCES

[1] Sander PA Alewijnse, Kevin Buchin, Maike Buchin, Andrea Kölzsch, Helmut Kruckenberg, and Michel A Westenberg. 2014. A framework for trajectory segmentation by stable criteria. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 351–360.

[2] Ove Andersen, Benjamin B. Krogh, and Kristian Torp. 2013. An Open-source Based ITS Platform. In *Proc. of MDM*, Vol. 2. 27–32.

[3] Adrian Balteanu, Gregor Jossé, and Matthias Schubert. 2013. Mining Driving Preferences in Multi-cost Networks. In *SSTD (Lecture Notes in Computer Science)*, Vol. 8098. Springer, 74–91.

[4] Maike Buchin, Anne Driemel, Marc Van Kreveld, and Vera Sacristán. 2011. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science* 2011, 3 (2011), 33–63.

[5] Paolo Campigotto, Christian Rudloff, Maximilian Leodolter, and Dietmar Bauer. 2017. Personalized and Situation-Aware Multimodal Route Recommendations: The FAVOUR Algorithm. *Trans. Intell. Transport. Sys.* 18, 1 (Jan. 2017), 92–102.

[6] Jian Dai, Bin Yang, Chenjuan Guo, and Zhiming Ding. 2015. Personalized route recommendation using big trajectory data. In *2015 IEEE 31st international conference on data engineering*. IEEE, 543–554.

[7] Daniel Delling, Andrew V. Goldberg, Moises Goldszmidt, John Krumm, Kunal Talwar, and Renato F. Werneck. 2015. Navigation Made Personal: Inferring Driving Preferences from GPS Traces. In *Proc. of SIGSPATIAL'15*. Association for Computing Machinery, New York, NY, USA, Article 31.

[8] Mohammad Etemad, Amílcar Soares Júnior, Arazoo Hoseyni, Jordan Rose, and Stan Matwin. 2019. A Trajectory Segmentation Algorithm Based on Interpolation-based Change Detection Strategies.. In *EDBT/ICDT Workshops*.

[9] Martin Fruensgaard and Tobias S. Jepsen. 2017. *Improving Cost Estimation Models with Estimation Updates and road2vec: a Feature Learning Framework for Road Networks.* Master's thesis. Aalborg University.

[10] Stefan Funke, Sören Laue, and Sabine Storandt. 2016. Deducing Individual Driving Preferences for User-Aware Navigation. In *Proc. of SIGSPATIAL'16*.

[11] Stefan Funke, Sören Laue, and Sabine Storandt. 2017. Personal Routes with High-Dimensional Costs and Dynamic Approximation Guarantees. In *16th Int. Symp. on Experimental Algorithms (SEA 2017) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 75. Dagstuhl, Germany, 18:1–18:13.

[12] Stefan Funke and Sabine Storandt. 2015. Personalized Route Planning in Road Networks. In *Proc. of SIGSPATIAL'15*.

[13] Tobias Skovgaard Jepsen, Christian. S. Jensen, and Thomas Dyhre Nielsen. 2020. Relational Fusion Networks: Graph Convolutional Networks for Road Networks. *IEEE Transactions on Intelligent Transportation Systems* (2020), 1–12.

[14] Tobias Skovgaard Jepsen, Christian S. Jensen, Thomas Dyhre Nielsen, and Kristian Torp. 2018. On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network. In *Proc. of Big Data 2018*. 3422–3431.

[15] Amilcar Soares Junior, Valeria Cesario Times, Chiara Renso, Stan Matwin, and Lucidio AF Cabral. 2018. A semi-supervised approach for the semantic segmentation of trajectories. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 145–154.

[16] Benjamin Krogh, Christian S Jensen, and Kristian Torp. 2016. Efficient in-memory indexing of network-constrained trajectories. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.

[17] Simon Aagaard Pedersen, Bin Yang, and Christian S Jensen. 2020. Fast stochastic routing under time-varying uncertainty. *The VLDB Journal* 29, 4 (2020), 819–839.

[18] Amílcar Soares Júnior, Bruno Neiva Moreno, Valéria Cesário Times, Stan Matwin, and Lucídio dos Anjos Formiga Cabral. 2015. GRASP-UTS: an algorithm for unsupervised trajectory segmentation. *International Journal of Geographical Information Science* 29, 1 (2015), 46–68.

[19] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. 2014. PRESS: A Novel Framework of Trajectory Compression in Road Networks. In *Proceedings of the VLDB Endowment*, Vol. 7. 661–672.

[20] Bin Yang, Chenjuan Guo, Yu Ma, and Christian S. Jensen. 2015. Toward Personalized, Context-aware Routing. *The VLDB Journal* 24, 2 (April 2015), 297–318.

[21] Yu Zheng. 2015. Trajectory Data Mining: An Overview. *ACM Trans. Intell. Syst. Technol.* 6, 3, Article 29 (May 2015), 41 pages.

APPENDIX

A TRAJECTORY STITCHING

Our trajectory dataset $D = \{\mathbb{T}_1, \ldots, \mathbb{T}_n\}$ consists of sets of trajectory sequences of the form $\mathbb{T}_i = (T_1, \ldots, T_m)$. Each trajectory sequence T_i contain trips specific to driver d_i and are in temporal order s.t. trajectory $T_j \in \mathbb{T}_i$ started before trajectory $T_{j+1} \in \mathbb{T}_i$. We define a stitched trajectory data set based on the data set D as $D_{stitched} = \bigcup_{i=1}^{n} \textsc{StitchTrajectories}(\mathbb{T}_i)$.

The StitchTrajectories function, defined in Algorithm 1, takes as input the trajectories of a driver in temporal order. We use $T_{current}$ to keep track of the current trajectory considered for stitching. Initially, $T_{current}$ is set to T_1. We use $e_{current}$ to keep track of the end time of the current trajectory $T_{current}$, i.e., its last recorded GPS point. In a loop, we scan the input trajectories $\mathbb{T}$ sequentially for

Algorithm 1 Trajectory Stitching

```
1:  function StitchTrajectories(𝕋 = (T_1, ..., T_n))
2:      𝕋_stitched ← ∅
3:      T_current ← T_1
4:      e_current ← GetEndTime(T_current)
5:      for i = 2 to n do
6:          s_i ← GetStartTime(T_i)
7:          if s_i − e_current ≤ 30 minutes and
8:                  T_i and T_current are pseudo-connected then
9:              T_current ← Stitch(T_current, T_i)
10:         else
11:             𝕋_stitched ← 𝕋_stitched ∪ {T_current}
12:             T_current ← T_i
13:         e_current ← GetEndTime(T_current)
14:     𝕋_stitched ← 𝕋_stitched ∪ {T_current}
15:     return 𝕋_stitched
```

stitching candidates, starting from trajectory T_2. We first store the time of the first GPS point associated with trajectory T_i in s_i.

We then check whether T_i is both temporally and spatially near enough to stitch with $T_{current}$. The two trajectories $T_{current}$ and T_i are temporally near enough to stitch if there is at most a 30 minute difference between $e_{current}$ and s_i. Two trajectories $T_1 = (e_1, \ldots, e_i)$ and $T_2 = (e_{i+1}, \ldots, e_{i+j})$, where $e_i = (u, v)$ and $e_{i+1} = (w, x)$, are pseudo-connected if the shortest route between v and w consists of at most one road segment or is less than 200 meters in length. If both stitching conditions are met, T_i is stitched to $T_{current}$ by invoking the Stitch function.

For pseudo-connected trajectories T_1 and T_2, Stitch is defined as $\textsc{Stitch}(T_1, T_2) = (e_1, \ldots, e_i, e'_1, \ldots, e'_k, \ldots, e_{i+j})$ where $(e'_1, \ldots, e'_k)$ is the shortest route connecting v and w which we refer to as a *stitch*. Then, the stitched trajectory is assigned to $T_{current}$.

If the stitching conditions are not met, we cannot stitch more trajectories to $T_{current}$. We then add the current trajectory to $\mathbb{T}_{stitched}$ and let T_i be the new current trajectory. Note that after the first iteration $T_{current}$ may be a stitched trajectory. For a stitched trajectory $T' = \textsc{Stitch}(T_1, T_2)$, we define $\textsc{GetEndTime}(T') = \textsc{GetEndTime}(T_2)$. After scanning through all of the input trajectories, we add the last trajectory to $\mathbb{T}_{stitched}$ and finally return the stitched trajectories.

B ROUTING COST TYPE DETAILS

In this section, we describe how the travel time, congestion, and crowdedness routing costs are derived in further details.

B.1 Travel Time

The vehicle trajectories in our trajectory set have the tendency to be concentrated on a few popular segments, as such, many road segments have few or no traversals in the trajectory set. We therefore require a means of estimating travel times for such road segments. To this end, we use a pre-trained machine learning model to provide travel time estimates. However, for road segments with an abundance of traversal data the model's estimates may be inaccurate. Inspired by previous work [9], we therefore combine travel time estimates with travel times of historical traversals s.t. when the driving speed estimate of a road segment becomes increasingly less influential the more historical traversals the road segment is associated with.

We compute the travel time t_e for a road segment e as $t_e = \frac{k\hat{t_e} + n\bar{t_e}}{k+n}$ where $\hat{t_e}$ is the estimate of the mean travel time, $\bar{t_e}$ is the mean travel time of the historical traversals, n is the number of historical traversals of segment e in the trajectory dataset, and k represents the confidence in $\hat{t_e}$. We use $k = 10$ in our experiments.

We use a pre-trained Relational Fusion Network (RFN) [13] to provide travel time estimates $\hat{t_e}$ for each road segment $e \in E$. Specifically, we use the best performing RFN from [13] which has been trained on the Danish Municipality of Aalborg using trajectories within the municipality that occurred between January 1st 2012 and June 30th 2013. Despite having been trained only on a subset of the network, the model generalizes well to unseen areas of the road network [13]. However, in a few cases the network would give very low values. We therefore modify the output s.t. the estimated driving speed on any road segment cannot be below 5 kmh.

B.2 Congestion

We assign a congestion level to road segment e depending on the speed limit s_e on the segment in km/h, the length of l_e of the segment in km, and the travel time t_e in hours. Let $\tau_e = l_e/s_e$ denote the travel time on road segment e if a vehicle is driving at exactly the speed limit. Formally, we assign road segment e the congestion level $c_e = \max\{1 - \frac{t_e}{\tau_e}, 0\}$ s.t. a value of 0 indicates that it is possible to drive at (or above) the speed limit and a value of 1 indicates that the road segment is not traversable.

The value of τ_e relies on the speed limit of road segment e. We use a speed limit data set that combines OSM speed limits with speed limits provided by Aalborg Municipality and Copenhagen Municipality [14]. This data set contains 163 044 speed limits, thus leaving many road segments without a known speed limit. In such cases, we use an OSM routing heuristic[3] which in Denmark assigns a speed limit of 130 km/h to motorways, a speed limit of 50 km/h in cities, and a speed limit of 80 km/h on other types of segments. For our data, we count a road segment as in a city if either the source or destination intersection is in a city according to its attributes.

B.3 Crowdedness

This routing cost type describes how 'crowded' the landscape around a road segment is. It is derived from the number of OSM nodes in the vicinity of the road segment. We use all OSM nodes in Denmark from a 2019 data set regardless wether they represent a road, a building or some other point of interest. To calculate it, we first overlay our graph with a grid and count the OSM nodes within each cell. For each road segment, we locate the OSM nodes that are part of its geometry in the grid. The cost per road segment is then the sum of the cell counts of its (geometry) nodes. We use a grid of 2000 by 2000 resulting in a cell size of roughly 209m x 177m.

[3] See https://wiki.openstreetmap.org/wiki/OSM_tags_for_routing/Maxspeed.