Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 170 (2020) 195–202

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 6 - 9, 2020, Warsaw, Poland

Identifying bicycle trip impediments by data fusion

Luk Knapen[a,c,*], Johan Holmgren[b,]

[a]*Hasselt University, Martelarenlaan 42, 3500 Hasselt, Belgium*
[b]*Internet of Things and People Research Center & Dept. of Computer Science and Media Technology, Malmö University, Malmö 205 06, Sweden*
[c]*VU Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands*

Abstract

A set of GPS traces for bicyclists and a set of notifications by bicyclists of problematic situations (spots identified by GPS records) had been collected independently. The data collection periods did not coincide but overlapped and none was contained in the other one. The aim is to use both datasets to determine an optimal action plan for problem solving given a limited budget.

First, problematic locations are clustered. Each cluster corresponds to an *impediment*. Impediments are then associated with trips using a distance function. The aim is to find out which impediments to solve under a given budget constraint in order to maximize the number of *impediment free* trips. Thereto the trip set is partitioned by matching each trip with the largest set of its affecting impediments. Solving all impediments in such set induces a cost and makes the associated part of trips impediment free. An optimizer is presented and evaluated.

Keywords: bicyclist; GPS traces; clustering; data fusion

1. Introduction

Two datasets have been collected in mutually independent projects. A set of GPS traces for 350 bicyclists has been recorded at high frequency (period between recordings in the range of [1 .. 5] seconds). A dataset containing timestamped GPS records (called clickPoints) has been recorded by 79 bicyclists during an overlapping period. Each clickPoint identifies a situation perceived by the bicyclist as problematic. Several clickPoints generated by multiple individuals at different times may identify a single *impediment* (e.g. a dangerous junction, a pavement problem, etc). The clickPoints identifying an impediment may have different coordinates. No information about the kind of impediment is supplied by the bicyclist. The datasets have been generated by participants in the two mutually independent projects (but it is not guaranteed that the respondent sets are disjoint).

* Corresponding author. Tel.: +32-11-25-47-21
E-mail address: luk.knapen@uhasselt.be

1877-0509
10.1016/j.procs.2020.03.025

An analyst is in charge of generating a semantic interpretation of the impediments using a GIS tool, estimating the cost to solve each impediment and advising which impediments to resolve given a budget constraint.

In order to support the analyst, impediments are identified by spatial clustering of intended clickPoints. Impediments are assumed to affect trips that pass nearby. We associate impediments with affected trips and find a set of impediments whose resolution makes as many trips as possible impediment free.

2. Related Work

Trajectory clustering is the focus of a lot of research (e.g. T-OPTICS and several variants). In many cases, trajectories are not considered to be network bound. Clustering is based on geometric properties of time series of GPS recordings. Many researchers [7, 21, 20, 1, 6] focus on trajectory annotation (mining of travel behaviour e.g. in order to establish route choice sets [15]). The authors of [9] use DBSCAN to find traffic stream clusters and define the concept of *cluster representative subsequence*. Others focus on applications where pairwise specific overlap of routes is required like in matching for carpooling [3]. In [18, 19] the authors present methods for *subtrajectory* clustering (and solve the problem of subtrajectory identification).

Few research applies *map-matching* prior to trajectory clustering; e.g. [8]) describes map matching based trajectory compression. Map-matching is not only interesting for storage of network constrained trajectories but also allows clustering based on link sequences. However map-matching is nontrivial [12, 16, 17, 11, 13, 2, 5, 4, 10] and is required only in particular applications.

Mobility related problem reporting is a hot topic. Using the search string "*app report problem to municipality*" in a browser delivers dozens of problem reporting apps aimed at notifying issues to the municipality government. Examples are (in alphabetical order) Box Clever (Canada), Cit2Adm (France), Colab (Brazil), FixMyCity (Germany), FixMyStreet (UK), iChangeMyCity (India), Improve My City (Greece), Link (South Africa), MobiMelder (Belgium), Munizapp (Iceland), OneService (Singapore), PublicStuff (USA) and probably more. Many allow to show the location of the issue on a map. Some are aimed at particular purposes (e.g. dedicated to pedestrians).

This paper aims to combine the results of *issue reporting tools* with GPS traces representing bicycle (regular and e-bike) trips. Thereto, clickPoints are spatially clustered using DBSCAN . Each cluster represents an impediment. In a preliminary research, k-means clustering was applied clustering to the same dataset of clickPoints [14].

Prior trajectory clustering using geometric properties is not applied. Each trip is processed individually and the combination (set) of impediments affecting the trip is determined. *Impediment combinations* then are used to partition the set of trips. This intermediate result is used as input for the optimizer.

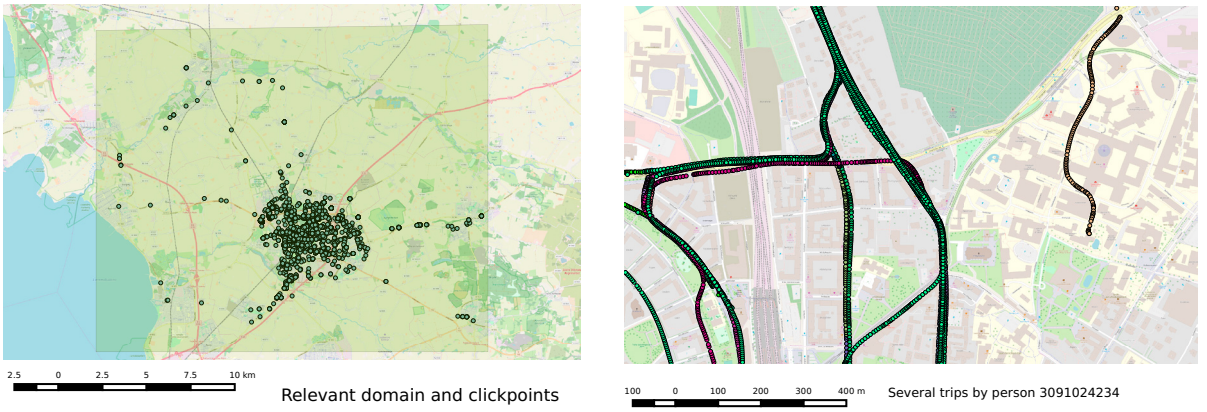
3. Data Properties

3.1. Notifications by Push Button: clickPoint

Data have been generated using a push button mounted on the bike and communicating to the bicyclist's smartphone using Bluetooth. A clickPoint record contains the *button identifier*, a *coordinate pair* generated by the smartphone, a *click-timestamp* and a *coord-timestamp*. The click-timestamp applies to the moment the button is pressed. The coord-timestamp applies to the GPS coordinates: if the coordinate pair cached in the smartphone is too old, a new pair is fetched; this sometimes takes a long time to complete. Records for which the difference between both timestamps is larger than 60[s] are discarded.

The difference between click-timestamps in consecutive records is often very small (less than 10[ms]); this is expected to stem from contact bounce. We assume that noise due to contact bounce only occurs for intended use of the button (hence does not emerge spontaneously e.g. due to vibrations caused by rough road pavement). We assume that a person does not press the button at a frequency higher than $1[Hz]$. For every individual, we consider a clickPoint with timestamp t to be an *intended clickpoint* (IntClickPoint) if and only if (i) it has no predecessor in the period $t - \Delta t$ or (ii) the duration of the noise sequence exceeds the value of the resolution parameter δ_{max}. In the experiments the values $\Delta t = 1[s]$ and $\delta_{max} = 5$[s] have been used. Contact bounce noise removal reduced the set of 3101 notifications to 2142 IntClickPoints. Figure 1a shows the IntClickPoints and the interactively selected *relevant domain* on a map.

Figure 1b shows trips for a particular individual.



(a) Relevant domain (shaded area) and clickPoints

(b) Several trips for a particular individual.

Fig. 1: Data overview: relevant domain, clickPoints and trips for a particular bicyclist.

3.2. Trips

Traces had been delivered as sequences of records specifying the smartphone identifier, the location and the timestamp. Data were recorded in the period 2018-Jul-11 to 2018-Oct-08 (included). Location data are not original raw GPS coordinates. Data seem to have been pre-processed by map-matching using fluent curves that approximate links on the underlying road network. The raw data clearly have been replaced by the smoothed coordinates. An example is shown in Figure 1b

The set of *relevant domain trips* consists of all bike mode trips that have at least one point in the relevant domain.

Trips have been extracted by splitting the traces using spatial and temporal thresholds. A trip is terminated at record r_i if and only if $\nexists r_{i+1} \vee \delta_e(r_i, r_{i+1}) > \Delta s \vee d_t(r_i, r_{i+1}) > \Delta t$ where $\delta_e()$ denotes the Euclidean distance and $d_t()$ denotes the time difference. In the experiments, $\Delta s = 1200[m]$ and $\Delta t = 120[s]$ were used.

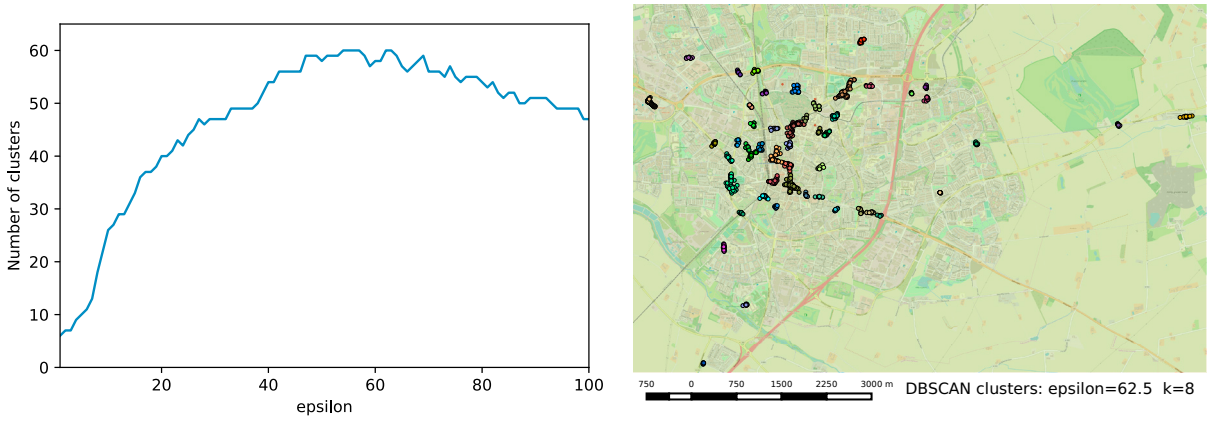
4. Associating Impediments and Trips

4.1. Impediments

An *impediment* is an obstacle, nuisance or problematic situation flagged by bicyclists generating IntClickPoints. An impediment may be associated to a road network *node* (junction) or *link* (road segment). Multiple IntClickPoints may be associated to a single impediment.

In this study it is assumed that the shape of an impediment can be stretched. Therefore, IntClickPoints are clustered using DBSCAN . Clustering by DBSCAN uses parameters ϵ (maximum distance between cluster members) and k (minimum cluster size). The following assumptions are used to determine the parameter values. An impediment has a length of $48[m]$, bicyclists drive at $6[m/s]$ in opposite directions and push the button not later than $3[s]$ after having passed the impediment. The distance between the cyclists when they push the button is at most $48 + 2 * 18 = 84[m]$. The values for ϵ used in the initial sensitivity analysis presented below are found in Section 6.1. The minimum cluster size is arbitrarily chosen to be $k = 8$. Note that IntClickPoints in a cluster may have been generated by one or more persons. Figure 2a shows how the number of clusters found by DBSCAN depends on ϵ for a minimum cluster size $k = 8$. Figure 2b shows the IntClickPoints belonging to identified clusters for $\epsilon = 62.5[m]$. Each cluster has its own color. Note that many clusters are *stretched* along road segments. Note the clusters near the city center: for $\epsilon = 100$ they merge into a single one.

Table 1 shows data for the first and last clusters found for $\epsilon = 50$. *Nr* is the cluster identifier. *nPts* is the number of IntClickPoints in the cluster (impediment). *nPrs* is the number of people who generated the IntClickPoints. Each



(a) Number of clusters as a function of ϵ for $k = 8$.

(b) Clustered IntClickPoints in the relevant area.

Fig. 2: DBSCAN clustering: the sensitivity analysis uses $\epsilon \in \{50,\ 62.5,\ 75\}$

Table 1: Sample clusters for $\epsilon = 50$. *Nr*: cluster number, *nPts*: number of IntClickPoints in the cluster, *nPrs*: number of people who clicked.

Nr	nPts	nPrs	Nr	nPts	nPrs	Nr	nPts	nPrs	Nr	nPts	nPrs	Nr	nPts	nPrs
0	26	10	1	17	9	2	69	16	3	71	14	4	21	7
5	18	8	6	22	8	7	14	6	8	10	2	9	8	3
10	15	3	11	18	6	12	15	1	13	46	4	14	8	2
15	18	5	32	9	3	33	11	2	34	13	1	35	8	4
36	11	6	37	8	5	38	12	2	39	8	5	40	10	4
41	69	1	42	10	1	43	8	1	44	9	6	45	10	2

DBSCAN cluster corresponds to an impediment. The number of clusters identified depends on the values for ϵ and k is reported in Table 2.

4.2. Association

Map matching was not used because it induces a two step method involving road network links (i.e. IntClickPoint-*link* and *link-traces* matching) and hence the risk to induce matching errors than can be avoided.

We are interested in clusters of trajectories that match in particular sets of regions each one corresponding to an impediment (as opposed to matching over the complete trajectory length).

When raw data are available, a distance threshold to determine whether or not a pair of GPS coordinates can have been produced from a given location. Thereto it is assumed that the GPS error obeys a Rayleigh distribution [2, 10]. The parameter for the distribution is determined by assuming that the observed error exceeds the specified device accuracy with a probability $p < 0.05$. However, because of the availability of pre-processed GPS traces, we choose $\Delta_s = 50[m]$ for the distance threshold based on visual inspection of the smoothed data.

The threshold Δ_s is used to associate an impediment (cluster of IntClickPoints) to a trip. Consider the set of pairs $\langle p_i^T, p_j^C \rangle$ where p_i^T is the coordinate pair for the i-th point in the trace T and p_j^I is the coordinate pair for the j-th IntClickPoint in the impediment. We consider T and I to be associated if and only if $| \{\langle p_i^T, p_j^I \rangle \mid d_e(p_i^T, p_j^I) \leq \Delta_s\} | \geq N$ where $d_e(\cdot, \cdot)$ denotes the Euclidean distance and $N = 3$. Note that both cases (i) "single trace point, multiple IntClick-Points" and (ii) "multiple trace point, single IntClickPoint" can occur.

5. Optimal Impediment Resolving under Budget Constraints

In order to make as many trips as possible impediment free under a given budget constraint, a method consisting of following steps is proposed.

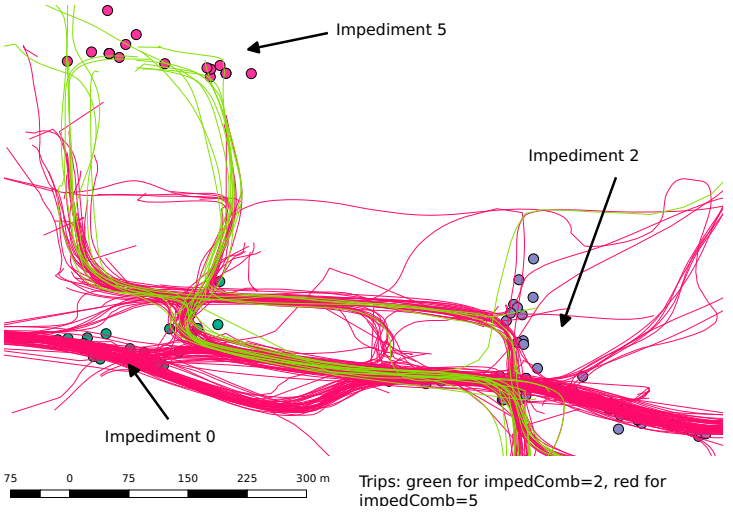
1. Impediments that have no associated trips are ignored.
2. The set of all impediments that apply to a particular trip T is called the *impediment combination* (impedComb) for T. Solving all impediments in an impedComb makes the associated trips *impediment free*. Some trips have no associated impediments: their impedComb=$\emptyset$.
3. For each trip the impedComb is determined. Consider for each impedComb K_i the set $T_i \in \mathcal{T}$ of trips that are affected by *exactly* the impediments in K_i; this is the *strict* trip set $T^S(K_i)$ for K_i. The set of impedCombs contains the empty set $\emptyset$ and defines a partition of $\mathcal{T}$ because each trip is associated with exactly one impedComb. The set of impedCombs and the parts constituting the partition are related by a bijection. The set of impedCombs is a small subset of the powerset $\wp(\mathcal{I})$ of the set of impediments $\mathcal{I}$.

   Figure 3a shows the *strict* trip sets for impedComb 2 (red, impediments $\{0, 2, 5\}$) and impedComb 8 (green, impediments $\{0, 2\}$). Both are parts in the partition of trip set $\mathcal{T}$ and hence disjoint. The number of impedCombs found for each case is reported in Table 2 for each ϵ value.
4. An acyclic digraph (the *smallest superset graph*) $G^H(V^H, E^H)$ is defined as follows. V^H is the set of impedCombs. Two impedCombs $K_i, K_j \in \mathcal{K}$ are connected by an edge if an only if $(K_i \subset K_j) \wedge (\nexists\, K_m \mid K_i \subset K_m \subset K_j)$. The superscript H is used because this is similar to a Hasse diagram. Graph G^H is used to compute the set of *all* trips affected by K_i (*complete* trip set) $T^K(K_i) = \bigcup_{k \in \mathcal{K} \mid \ell(k,K_i)} T^S(k)$ where $\ell(k, K_i)$ indicates that in G^H vertex k is on a path connecting K_i to a leaf vertex. Note that $\neg(K_i \neq K_j \Rightarrow T^K(K_i) \cap T^K(K_j) = \emptyset)$ (the *complete* trip sets of different impedCombs may overlap).
   The cost $C(K_i)$ to solve all impediments in the impedComb K_i is given by $C(K_i) = \sum_{I \in K_i} C(I)$.
5. The objective is to determine the set of impediments for which the total cost is below a given budget threshold and that maximizes the impediment free trips when resolved. Note that solving a strict subset of an impedComb K_i is only useful if $\exists K_j \subset K_i$ because any part $T(K)$ of the trip set partition is made impediment free only if all impediments in K are solved. Hence, the basic concept in the optimization is the impedComb.
6. Because not all impedComb pairs are disjoint, the cost to resolve an impedComb and the gain (the number of trips made impediment free) depend on which impedComb have been (partially) resolved before. Hence, the impedComb intersection graph is used to guide the computation. The optimizer is a recursive algorithm that consecutively considers sets of impediments to *resolve* or to *block*. Obviously, the corresponding *undo actions* occur during backtracking. The states for an *impediment* then are: *waiting* (for resolution), *resolved* and *blocked* (for resolution). The states for an *impedComb* are nominally similar.

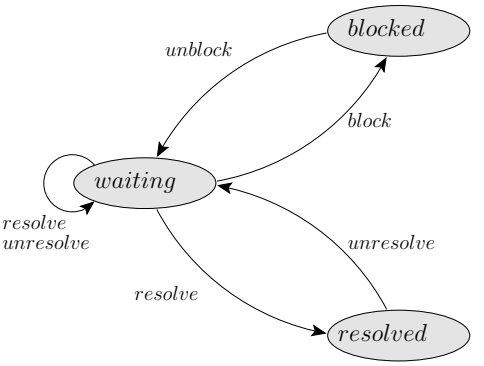
$$state = \begin{cases} blocked & if \quad at\ least\ one\ of\ it\ elements\ is\ blocked \\ resolved & if \quad all\ impediments\ are\ resolved \\ waiting & else \end{cases} \tag{1}$$

The state transition diagram is shown in Figure 3b

The optimizer uses two main structures: (i) an ordered list of impedCombsorted by decreasing *complete affected trip set* size (ii) the intersection graph for impedCombs It basically uses recursion to enumerate solutions by finding the first non-blocked impedComb in the sorted list for which the resolution is within budget and then considers both the *resolve* and *block* actions. The intersection graph is used to update the state of each impedComb based on the selected action. Finally, this is repeated recursively and the action is undone. The use of the intersection graph aims to minimize the computational cost to maintain the state.



(a) Trips for two impedCombs



(b) State transition diagram for impedComb

Fig. 3: impedCombs : Associated trips - State transition diagram

Resolving an impedComb K_i resolves all impedCombs in the containment diagram on each path connecting a leaf vertex to K_i. This does not hold for *blocking* because blocking a single impedimentI_m blocks K_i if $I_m \in K_i$ but does not necessarily block all $K_j \subset K_i$. On the other hand, blocking an impediment blocks all impedCombs it is contained in. The effectiveness of the mentioned operations and the high density of the intersection graph (e.g. 0.176 for the $\epsilon = 50$ case) make enumeration feasible for practical cases.

6. Results - Discussion

6.1. Results

1. In reality cost to solve impediment is determined by user. For evaluation of the technique we used a *dummy* cost specified by $c(i) = \mid i \mid \cdot \max_{q \in i} n(q)$ where i denotes the impediment, $\mid i \mid$ is the impediment size (number of IntClickPoint) and $n(q)$ is a IntClickPoint number. The cost range for the $\epsilon = 50$ case is [16448 , 258795]. The other cases show similar ranges. python3 and postgresql 11 were used.

2. The experiments have been carried out on a Debian Linux 10.2 (Buster) on a Pentium(R) Dual-Core CPU E5200 @ 2.50GHz and the optimizer was allowed to run for at most 1200 seconds but that limit was never exceeded.

3. The number of trips having at least one point in the relevant domain is 22953.

4. Two parameters have been varied for a first sensitivity analysis. The maximum distance ϵ to the nearest neighbour for each IntClickPoint in DBSCAN clusters was set to 40, 50, 60, 70 and 80 meters respectively. The available budget to resolve impediments was set to $1M$, $2M$ and $3M$ respectively. The results are summarized in Table 2: (i) *nTrips* is the number of trips made impediment free, (ii) *nImped* is the number of impediments to solve (they are not all listed in the table), (iii) *cost* is the amount required to solve the impediments and (iv) *searchTime* is the optimizer execution time in seconds.

6.2. Discussion and future work

The data preparation to associate trips with IntClickPoints takes 1466[sec] (much more than the optimizer). However, this step needs to be performed only once and is *embarrassingly parallel*.

Table 2: Results summary for first sensitivity analysis: (U) absolute numbers in upper part, (M) relative numbers (fractions) in middle part, (L) cost and time [sec] after which no better solution was found in the lower part. *# Imp* the number of impediments *# ImpC* is the number of impedCombs. *Cost* is the total cost to solve all impediments. *# AffT* is the number of trips affected by impediments.

	ϵ = 40 [m]		ϵ = 50 [m]		ϵ = 60 [m]		ϵ = 70 [m]		ϵ = 80 [m]	
	nTrips	nImped	nTrips	nImped	nTrips	nImped	nTrips	nImped	nTrips	nImped
Budget 1*M*	3454	13	2537	11	2879	13	2556	13	2284	9
Budget 2*M*	7593	37	6561	31	5880	29	5082	24	4095	13
Budget 3*M*	8390	46	8688	46	9134	47	8549	42	7290	30
Budget 1*M*	0.411	0.282	0.292	0.239	0.312	0.265	0.265	0.254	0.233	0.187
Budget 2*M*	0.905	0.804	0.755	0.673	0.639	0.591	0.527	0.470	0.419	0.270
Budget 3*M*	1.000	1.000	1.000	1.000	0.992	0.959	0.886	0.823	0.746	0.625
	cost	dur[s]	cost	dur[s]	cost	dur[s]	cost	dur[s]	cost	dur[s]
Budget 1*M*	991306	0.028	995102	0.015	997303	0.008	999460	0.007	993379	0.006
Budget 2*M*	1998484	0.016	1986900	36.155	1999024	1.109	1994177	0.165	1999989	0.059
Budget 3*M*	2400405	0.020	2779005	0.076	2969726	0.022	2988972	0.597	2997958	1.217
	# Imp	# ImpC	# Imp	# ImpC	# Imp	# ImpC	# Imp	# ImpC	# Imp	# ImpC
	46	823	46	846	49	978	51	1047	48	922
	Cost	# AffT	Cost	# AffT	Cost	# AffT	Cost	# AffT	Cost	# AffT
	2400405	8390	2779005	8688	3129388	9199	3411246	9636	3729974	9771

In general exhaustive enumeration is never guaranteed to run in a decent time. The problem, although not trivial and using real data, was solved in very short time in most of the cases. In case budget=2M, ϵ = 50 the best solution was found after 0.504[*s*] but it is obvious that the solution is known to be optimal only after 36[*s*] when the enumeration is complete. Therefore, further research will focus on a finding tight upper bounds for the potential gain in order to support a branch and bound method.

The optimizer is sufficiently fast to be used in interactive optimization sessions where the analyst has to determine the cost to solve the impediments.

7. Conclusion

An optimization method to support the analyst when selecting impediments to solve under a budget constraint. Impediments are determined by clustering IntClickPoints. The sensitivity analysis shows the effect of the selection of the parameter ϵ in DBSCAN . If the budget is much lower than the total resolution cost (*1M* case), the number of impediment free trips decreases monotonically with ϵ. In the case the budget is comparatively large (*3M* case) the optimum occurs in the range where ϵ maximizes the number of clusters. The proposed technique is sufficiently fast for interactive use. This is important because impediment solution cost estimation is *iterative* and *interactive*.

Nomenclature

clickPoint	Notification (location, time) from button operated by bicyclist
IntClickPoint	Intended clickPoint (intentionally generated) determined by filtering contact bounce noise
Impediment	Set of spatially clustered IntClickPoints
impedComb	Impediment combination: maximal set of impediments affecting at least one trip

Acknowledgements

The research leading to this paper was partially supported by the *Smarta Offentliga Miljöer II (SOM II)* project of the Lund (Sweden) municipality by supplying data.

References

[1] Andrienko, G., Andrienko, N., Hurter, C., Rinzivillo, S., Wrobel, S., 2011. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data, in: IEEE Conference on Visual Analytics Science and Technology, IEEE, Providence, Rhode Island, USA.

[2] Bierlaire, M., Chen, J., Newman, J., 2013. A probabilistic map matching method for smartphone {GPS} data. Transportation Research Part C: Emerging Technologies 26, 78 – 98. URL: http://www.sciencedirect.com/science/article/pii/S0968090X12001064, doi:10.1016/j.trc.2012.08.001.

[3] Cruz, M.O., Macedo, H., Guimar£es, A., 2015. Grouping Similar Trajectories for Carpooling Purposes, in: 2015 Brazilian Conference on Intelligent Systems (BRACIS), pp. 234–239. doi:10.1109/BRACIS.2015.36.

[4] Deka, L., Quddus, M., 2015. Trip-Based Weighted Trajectory Matching Algorithm for Sparse GPS Data, in: TRB 94th Annual Meeting Compendium of Papers, TRB (Transportation Research Board), Washington, D.C.

[5] Deng, Z., Hu, Y., Zhu, M., Huang, X., Du, B., 2014. A scalable and fast OPTICS for clustering trajectory big data. Cluster Computing 18, 549–562. doi:10.1007/s10586-014-0413-9.

[6] Furletti, B., Cintia, P., Renso, C., Spinsanti, L., 2013. Inferring human activities from GPS tracks, in: UrbComp 13 Proceedings of the second ACM SIGKDD International Workshop on Urban Computing, ACM, Chicago.

[7] Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D., 2007. Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA. pp. 330–339. URL: http://doi.acm.org/10.1145/1281192.1281230, doi:10.1145/1281192.1281230.

[8] Kellaris, G., Pelekis, N., Theodoridis, Y., 2013. Map-matched trajectory compression. Journal of Systems and Software 86, 1566 – 1579. URL: http://www.sciencedirect.com/science/article/pii/S0164121213000289, doi:10.1016/j.jss.2013.01.071.

[9] Kim, J., Mahmassani, H.S., 2015. Spatial and Temporal Characterization of Travel Patterns in a Traffic Network Using Vehicle Trajectories. Transportation Research Procedia 9, 164 – 184. URL: http://www.sciencedirect.com/science/article/pii/S2352146515001702, doi:10.1016/j.trpro.2015.07.010.

[10] Knapen, L., Bellemans, T., Janssens, D., Wets, G., 2018. Likelihood-based offline map matching of {GPS} recordings using global trace information. Transportation Research Part C: Emerging Technologies 93, 13 – 35. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18307022, doi:10.1016/j.trc.2018.05.014.

[11] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y., 2009. Map-matching for Low-sampling-rate GPS Trajectories, in: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, New York, NY, USA. pp. 352–361. URL: http://doi.acm.org/10.1145/1653771.1653820, doi:10.1145/1653771.1653820.

[12] Marchal, F., Hackney, J., Axhausen, K.W., 2005. Efficient Map Matching of Large Global Positioning System Data Sets: Tests on Speed-Monitoring Experiment in Zuerich. Transportation Research Record: Journal of the Transportation Research Board 1935, 93–100. doi:10.3141/1935-11.

[13] Ochieng, W.Y., Quddus, M., Noland, R., 2010. Map-Matching in Complex Urban Road Networks. Revista da Sociedade Brasileira de Cartografia, Geodsia, Fotogrametria e Sensoriamento Remoto 55, 14.

[14] Persson, M.A., Olsson, V., 2019. Cyclists perceived insecurity in urban environment - An unsupervised machine learning study. Bachelor Data Science. Malmo University. Malmo, Sweden.

[15] Pillat, J., Mandir, E., Friedrich, M., 2011. Dynamic Choice Set Generation Based on Global Positioning System Trajectories and Stated Preference Data. Transportation Research Record 2231, 18–26. URL: http://dx.doi.org/10.3141/2231-03, doi:10.3141/2231-03.

[16] Quddus, M.A., Ochieng, W.Y., Noland, R.B., 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation Research Part C: Emerging Technologies 15, 312 – 328. URL: http://www.sciencedirect.com/science/article/B6VGJ-4P2JCX9-1/2/21d1ef73e42329087fc872e39ce78b1b, doi:10.1016/j.trc.2007.05.002.

[17] Schssler, N., Axhausen, K.W., 2009. Map-matching of GPS traces on high-resolution navigation networks using the Multiple Hypothesis Technique (MHT). Working Paper 589. ETH Zrich. Zrich.

[18] Tampakis, P., Doulkeridis, C., Pelekis, N., Theodoridis, Y., 2019a. Distributed Subtrajectory Join on Massive Datasets. URL: https://arxiv.org/abs/1903.07748.

[19] Tampakis, P., Pelekis, N., Doulkeridis, C., Theodoridis, Y., 2019b. Scalable Distributed Subtrajectory Clustering. URL: https://arxiv.org/abs/1906.06956.

[20] Yan, Z., Spaccapietra, S., 2009. Towards Semantic Trajectory Data Analysis: A Conceptual and Computational Approach, in: VLDB2009 (PhD Workshop).

[21] Zheng, Y., Chen, Y., Li, Q., Xie, X., Ma, W.Y., 2010. Understanding Transportation Modes Based on GPS Data for Web Applications. ACM Trans. Web 4, 1:1–1:36. URL: http://doi.acm.org/10.1145/1658373.1658374, doi:10.1145/1658373.1658374.