Robust and Hierarchical Stop Discovery in Sparse and Diverse Trajectories

Le Hung Tran *
HoChiMinh City Univ. of Tech.
HoChiMinh, Vietnam
hung.tranle@epfl.ch

Nguyen Quoc Viet Hung
EPFL
Lausanne, Switzerland
quocviethung.nguyen@epfl.ch

Ngoc Hoan Do *
HoChiMinh City Univ. of Tech.
HoChiMinh, Vietnam
hoan.do@epfl.ch

Zhixian Yan
EPFL
Lausanne, Switzerland
zhixian.yan@epfl.ch

***Abstract*—The advance of GPS tracking technique brings a large amount of trajectory data. To better understand such mobility data, semantic models like "stop/move" (or inferring "activity", "transportation mode") recently become a hot topic for trajectory data analysis. Stops are important parts of trajectories, such as "working at office", "shopping in a mall", "waiting for the bus". There are several methods such as *velocity*, *clustering*, *density* algorithms being designed to discover stops. However, existing works focus on well-defined trajectories like movement of vehicle and taxi, not working well for heterogeneous cases like diverse and sparse trajectories.**

On the contrary, our paper addresses three main challenges: (1) provide a robust clustering-based method to discover stops; (2) discover both *shared stops* and *personalized stops*, where *shared stops* are the common places where many trajectories pass and stay for a while (e.g. *shopping mall*), whilst *personalized stops* are individual places where user stays for his/her own purpose (e.g. *home, office*); (3) further build stop hierarchy (e.g. a big stop like *EPFL campus* and a small stop like *an office building*). We evaluate our approach with several diverse and spare real-life GPS data, compare it with other methods, and show its better data abstraction on trajectory.

I. Introduction

In recent years, there have been a tremendous surge in applications and services with location feeds. This is possible in turn due to the large-scale embedding of GPS equipped mobile devices. There are emerging many relevant real-life applications. To mention a few: (1) scientists implant GPS chips in animals to analyze the social behavior of wild life, e.g. bird migration or monkey habits in forest. (2) smart phones (e.g. iPhone, Nokia N series) can help people establish geo-social networks and provide location-based services. (3) RFID technology installed in goods can improve e-business with better tracking of shipment. (4) GPS-furnished moving vehicles can enhance real-time traffic analysis and provide better road planning to decrease or even avoid congestion.

Therefore, trajectories become ubiquitous in many diverse applications, and grow to be a huge data source as tracking time goes by. To better understand such mobility data, many data mining techniques have been applied in data abstraction and discovering interesting mobility patterns, including techniques such as clustering [11], classification [10], outlier detection [12], finding convoys [8] and sequential rule-driven pattern mining [7], over real-life GPS data feeds.

* Currently working in EPFL as visiting student.

In this paper, we explore the problem of *stop discovery* in trajectories. Stops are the important places where trajectory has passed and stayed for a while. Taking Fig. 1 for instance: the green dots show the original GPS points that trajectory recorded, i.e. the evolution of physical position as time grows, as a sequence of spatio-temporal $\langle x, y, t\rangle$ points; the four circles show the important places where trajectory has stayed. With the result of such stop discovery method, we can explain the trajectory in a more meaningful way instead of the initial GPS $\langle x, y, t\rangle$ trace: *the tracking user started from home, went to EPFL for work, after off-duty he did shopping in COOP for a while, and finally reached Home*.

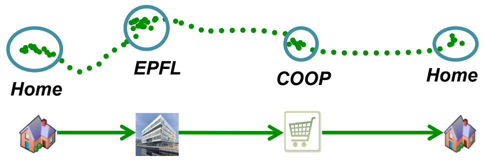


Fig. 1: Important places (*stops*) in trajectory

The notation of *stop* can be traced back to a conceptual view on trajectory proposed in [15], where trajectory is composed of a *begin* and an *end*, together with a sequence of alternative *stops* and *moves*. Several forthcoming literatures study algorithms in computing such stops/moves like [2][14][21][18]. This paper focuses on providing an efficient and robust stop discovery method, which can work well for different kind and quality of trajectory datasets of diverse moving objects.

A. Benefits of Stop Discovery

For a better data abstraction on the original mobility trace (i.e. the initial sequence of GPS points $\langle x, y, t\rangle$), we need to divide it into a sequence of episodes, and pick up the important episodes – *stops*. There are several remarkable features for the notation of *stops* in trajectory data abstraction:

- *Easily understandable*: A sequence of stops can provide a better abstracted view for understanding mobility trace, rather than the original sequence of $\langle x, y, t\rangle$ points. As shown in Fig. 1, only important places (i.e. *home*, *EPFL* and *Coop*) need to be highlighted for that trajectory.
- *Efficient data compression*: Instead of keeping the whole mobile tracking points, mobility data can be represented and restored in terms of a sequence of stops. As shown in

Fig. 1, more than hundred points can be compressed and displaced by four stops, actually only three stop places.

- *Automatic stop computation*: These important parts of trajectories (*stops*) can be computed automatically and efficiently, based on the relevant trajectory data discretization/segmentation methods, as the focus of this paper.

The original definition of *"stop"* was emphasized as "not moving" [15]. However, real-life stops are not purely depending on whether it is moving or not. For example, short time of stilling like a congestion in the middle of a road is not a true stop; shopping with wondering around in a store can be considered as a stop even the user is moving. Therefore, we define stops as *the important places where a trajectory has passed and stayed for a reasonable time duration*. Stops can help people summarize trajectory and provide better understanding. We thus do not distinguish among "stops", "hotspots", "important/meaningful/interesting parts" of trajectories etc. The focus is on designing robust algorithms to automatically discover stops in heterogeneous trajectories of diverse moving objects.

B. Additional Stop Characteristics

Different from existing studies on stop computation in the literature [2][14][21][18], our paper targets at a robust and efficient stop discovery algorithm, which can explore more challenging issues with additional characteristics of stops.

1) Shared Stops vs. Personalized Stops: Shared stops are the common places where many trajectories of different moving objects pass and stay for a while (e.g. *shopping mall* like COOP), whilst *personalized stops* are individual places where user stays for his/her own purpose (e.g. *home*). For example, there are three trajectories in Fig. 2, where each one has 4 stops. In addition, there are three common places including *EPFL*, *Coop*, and *SportCenter* being correlated by these trajectories as shared stops; whist the other 6 stops are personalized stops belonging to each trajectory separately, could be individual *home* for instance.

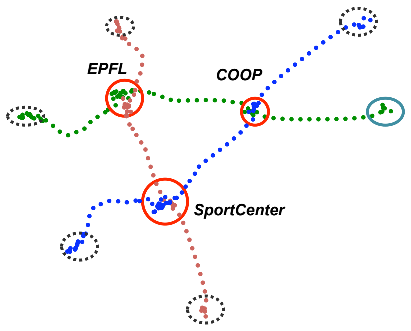


Fig. 2: Shared Stops vs. Personalized Stops

Therefore, our stop discovery algorithm will consider both shared stops and personalized stops – distinguishing them. Personalized stops usually can be discovered from single trajectory, whist shared stops can be extracted from a group of trajectories belonging to different moving objects.

2) Hierarchical Stops (Generic Stops vs. Concrete Stops): Another challenging problem is the granularity of the stops, i.e. determining the suitable stop size. Stop discovery in trajectory might be meaningless when it produces too many stops or too few stops. Each stop should have a suitable size, which means it includes a reasonable set of GPS points with very high correlations. Some applications require large stops whilst others may prefer smaller ones.

- *Generic Stops*: Taking study in EPFL for example, a student came to *Classroom1* for taking the Database course, and then left for the Algorithm course in *Classroom2*. Precisely, there are two small stops in different classrooms. However, application might be more interested in the generic stop (i.e. EPFL) when analyzing student daily behavior, where concrete classroom is nonsignificant.
- *Concrete Stops*: In contrast, some applications care more about concrete stops rather than generic ones. E.g. people visit a big commercial centre, which has many interesting places such as restaurant, supermarket, cinema etc. For trajectory applications, commercial center is too generic. Applications want to know exactly which place people visited. Therefore, one large stop needs to be split into many concrete and meaningful ones.

To classify stops into *generic* and *concrete* ones, we design a *hierarchy-based* stop discovery approach. The stops can be represented in different levels. The stop in lower level is in small size with more detailed information; whilst stops in higher level are usually big and with only generic positioning. Consequently, trajectories can be abstracted in terms of stop sequences of different levels. Fig. 3 sketches the refinement of discovered stops (left), in terms of tree-based stop hierarchy established (right), based on a *split-merge* approach.

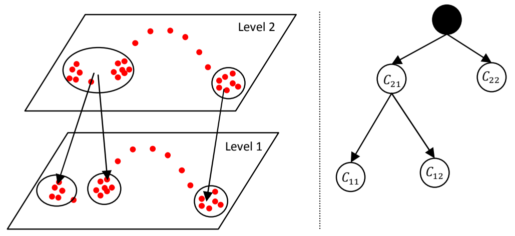


Fig. 3: Building tree-based hierarchal *stops*

C. Stop Discovery in Heterogeneous Trajectories

Existing works like [2][14][21][18] on discovering stops focus on homogeneous trajectories, which means there are a couple of universal requirements of such GPS datasets: (1) the moving object is identical like a taxi, a truck, or a ship, therefore the speed/acceleration of trajectory is more or less stationary; (2) the GPS sampling frequency is stationary, e.g. the time gap between the continuous GPS points is almost fixed (e.g. every one second or minute). In addition, these stop discovery algorithms do not work properly for GPS data with big recording gaps. However, the real-life trajectories are far more heterogeneous. Taking people trajectories from smartphones for instance, GPS from such people-centric tracking devices are much heterogeneous with ambiguity, e.g. (1) people can take many different kinds of transportation modes, such as *walking*, *cycling*, *driving*, *public transports like metro or bus* etc., therefore, GPS of people trajectories are much more irregular compared with that from taxi or truck. (2) due to GPS signal loss during people indoor activities[1] and the limited power issue of smartphones, GPS sampling of people trajectories are very sparse. Therefore, the objective of our stop discovery algorithms is to work robustly for the heterogeneous trajectory datasets as well.

II. Related Work

The literature on stop discovery algorithms can be divided into two categories, i.e. *static* and *dynamic*.

For the *static* approach, the physical positions of important places such as *EPFL campus*, *COOP* shopping mall in Fig.1 are given in advance to compute their spatial intersections with trajectories. These geographic places are usually called *Point Of Interests* (POI) or *Region/Line of Interests* (ROI/LOI) depending on their underlying geometric shape [17]. Such data can be extracted from relevant 3rd party geographic data sources like Openstreetmap[2]. For example, the authors of [2][16] have applied POI data in computing stops to enrich trajectories and even infer activities. Typically, the static methods firstly compute candidate stops based on the spatial joins between trajectory and POIs/ROIs, and then apply additional stop constrains (e.g. *time duration*) to filter out non-meaningful stop candidates. Stop discovery with the static approach is limited to geographic data that intersect the trajectories for a certain time interval. It is interesting for applications when the important POI data is available in the area of trajectories. However, the difficulty in getting such well-defined POI data significantly limit these static methods.

More literatures study the *dynamic* solution, where no POI data is available in advance, and stops only can be discovered according to trajectory's spatio-temporal characteristics, such as *velocity*, *acceleration*, *orientation* etc. Certain threshold-based methods are provided for determining stops. For example, Agamennoni et al. design a score function to create cluster as significant locations by a fixed speed threshold [1]; Yan et al. design a computing platform to identify stops according to a dynamic velocity threshold (Δ_{speed}), where continuous GPS points with velocity below Δ_{speed} are grouped together as stops, otherwise as moves [18]. Zheng et al. apply two fixed thresholds (distance D_{threh} and time T_{threh}) to detect so called "stay point" - actually the similar notation as stop [19]. In such threshold-based methods, the size and number of resulting stops are quite sensitive to the thresholds.

Additionally, another big branch of *dynamic* approach is applying density methods for discovering stops based on well-known clustering methods (e.g. k-Mean [13], DBSCAN [6], OPTICS [3]) like [4][20][14][21][9][5]. Ashbrook et al. in [4] use k-Mean to learn important locations from historical GPS data, by dividing the location trace into k pieces and then identifying the important pieces. Obviously, it is non-trivial to provide a fix k value. Both authors in [20] and [14] refine the DBSCAN algorithms for finding stops, where the focus of [20] is on efficiency by decreasing the memory requirement whilst [14] emphasizes on the additional time constrains based on the clustering on the speed of moving object. Zimmermann et al. extend OPTICS for discovering stops in [21], where the authors define the additional trajectory distance of two GPS points and trajectory neighborhood in judging whether a point is a core point in OPTICS. Recently, Kami et al. in [9] apply density-based random sampling based on the histograms of Locality Sensitivity Hashing (LSH), inferring stops with probabilistics. Cao et al. in [5] design "Semantic-Enhanced Clustering (SEM-CLS)" by using geocode to determine whether stay points need to be merged or split – as a stop point, and claim such method can provide better results compared to k-Means and OPTICS.

The above methods, both static and dynamic ones, have made significant progress in finding stops or so called important places in trajectories. However, there are a couple of drawbacks like (1) using fixed parameters [1][19]; (2) requiring and depending on additional geographic sources [2][16][5]; (3) there are works [5] [20] mentioning split/merge or hierarchy, but none of them consider computing stops at different granularity; (4) all the algorithms are in the context of stationary GPS datasets (As discussed in Section I-C). On the contrary, our paper is aiming at a more robust algorithm on stop discovery in sparse and noisy trajectories, and provide further stop analysis such as stop hierarchy, shared/personalized stops. The core contributions of our paper include the following three aspects:

- provide a robust clustering based approach to discover stops in different kinds of trajectory datasets, validating its efficiency compared with different methods;
- discover both *shared stops* and *personalized stops*, where *shared stops* are the common places where many trajectories pass and stay for a while (e.g. *shopping mall*), whilst *personalized stops* are individual places where user stays for his/her own purpose (e.g. *home, office*);
- analyze stops in a hierarchal sense (e.g. a big stop like *EPFL campus* and a small stop like *an office building*), which can support different levels of stop granularity.

III. Problem Statement

The main research problem of this paper is: *how to abstract/discretize trajectory in terms of stop discovery?* In details, we will design relevant clustering-based algorithms to automatically discover important parts of trajectories, where each part is a separate episode named *stop* for easily and semantic mobility data understanding. Therefore, the objective of this paper is to construct high-level semantic trajectories (i.e. *a sequence of stops*) from low-level raw trajectory data (i.e. *a sequence of GPS records*), see Fig. 1.

[1]From Nokia report, people spent nearly 80% of their time indoor without GPS recordings.

[2]http://www.openstreetmap.org/

The raw GPS-based trajectory is composed of a sequence of spatio-temporal points, i.e. $\langle x, y, t\rangle$ collected by GPS-alike mobile devices (see Def. 1). The spatio-temporal point $\langle x, y, t\rangle$ contains the evolution of physical position (by *longitude* x and *latitude* y) in geographic coordinate system. With the tracking time goes by, the sequence of GPS feeds grows continuously and the size can be huge.

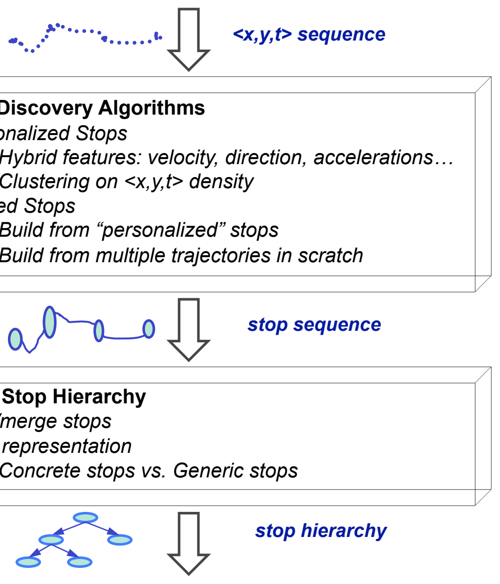
Definition 1. *Trajectory* $(\mathcal{Q})$ *- A sequence of spatio-temporal points recording the trace of a moving object, i.e.* $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$, *where* $Q_i = (x, y, t)$ *is a tuple including a position* (x, y) *and a timestamp* (t).

The "stop discovery" problem can be formally described as one kind of trajectory segmentation problem, where the trajectory is divided into a series of "episodes" [18]. The spatio-temporal points in a single episode is more correlated than the points outside. Episode can be marked as "stop","move", "begin","end","loop" etc., according to their underlying motion characteristics like "velocity", "acceleration" etc. In this paper, we consider "stops" as "important places" of trajectories, and need to be particularly extracted.

Definition 2. *Stop Discovery - Extract important parts ("stops") of trajectory, where each stop is a subpart of the original trajectory* (x, y, t) *sequence. A stop can be further summarized as* $\mathcal{S} = (\mathcal{L}, \Delta t_{min}, \Delta t_{max})$, *where* $\mathcal{L}$ *is the location and* Δt_{min} (Δt_{max}) *is the minimal* (*maximal*) *time duration when the trajectory stays in this location.*

As previously mentioned, we exploit the three important issues regarding "stop discovery" in trajectories, i.e. (1) *robust stop discovery algorithm in heterogenous trajectory dataset*, (2) *discover both personalized and shared stops*, and (3) *build the stop hierarchy considering both generic and concrete stops*.

To cover these issues, we build the following *stop discovery* framework (see Fig. 4). The initial input is the trajectory data in terms of (x, y, t) records; the first major component is *Stop Discovery Algorithms* on finding both personalized and shared stops via different methods; afterward, we design algorithms for spit/merge stops in the second component, building tree-based stop hierarchy (for generic and concrete stops) and further summarizing the important places in trajectories.



<x,y,t> sequence
Stop Discovery Algorithms
-Personalized Stops
•Hybrid features: velocity, direction, accelerations…
•Clustering on <x,y,t> density
-Shared Stops
•Build from "personalized" stops
•Build from multiple trajectories in scratch
stop sequence
Build Stop Hierarchy
- split/merge stops
- stop representation
•Concrete stops vs. Generic stops
stop hierarchy

Fig. 4: The framework for *stop discovery*

IV. Stop Discovery Algorithms

As described in Section II, static approach on stop discovery have several drawbacks, therefore we focus on dynamic approach without using any predefined geographic information.

In trajectory applications, stops can be shared stops or personalized stops (see the additional stop characteristics in Section I-B). In this section, we present the detailed algorithms that can discover such two kinds of stops separately and correlatively. Subsection IV-A focuses on inferring personalized stop; and Subsection IV-B is for discovering shared stops.

A. Personalized Stop Discovery

We analyze two different methods for finding personalized stops: (1) based on the features of spatio-temporal points, such as velocity, direction, acceleration etc. (2) clustering method based on the pure $\langle x, y, t\rangle$ GPS points.

1) Feature-based Stop Discovery: Many stop discovery methods are built on analyzing one or several features of the spatio-temporal point $\langle x, y, t\rangle$ of trajectory such as *spatial distance to previous point*, *time gap to previous point*, *velocity*, *acceleration*, *direction*, etc. [1][18][19].

The first common feature is "velocity". The velocity in stops is usually lower than other parts of trajectory. Therefore, stop can be discovered by using a speed threshold (Δ_{speed}), that is, if the instant speed of a tuple is lower than the threshold, it can be considered as a part of a stop, and vice versa [18]. The problem is the uncertainty in determining the right value for Δ_{speed}. If Δ_{speed} is too high, many stops will appear; on the contrary, if Δ_{speed} is too low, probably no stop will be computed. Therefore, choosing a good Δ_{speed} is challenging. Some candidate solutions are: choosing Δ_{speed} based on external knowledge, e.g. the average human walking speed (1.2m/s), the average speed of the whole trajectory, the current vehicle type and environment etc. In addition, to avoid force positive like congestion with low speed in a short term, an extra threshold is needed – *minimal stop time* (see [18]). Velocity-based approach is useful for applications in which speed plays an essential role, such as traffic management.

Similar to velocity, "acceleration" in stops is usually lower than other parts. The acceleration often decrease (as negative) at the beginning of stops and increase (as positive) at the end of stops. However, moving objects tend to change acceleration irregularly, and acceleration is an uncertain feature of trajectory as it is much non-stationary compared with velocity and strongly depends on other information such as the type of vehicle, road condition, etc. Thus, the acceleration can be considered as a support feature in stop discovery process, particularly with velocity together.

Yet another feature is "direction". When moving, the direction of moving objects is not change regularly; on the contrary, moving objects in stops have tendency to change its direction with high frequency and wide angle. Stop can be discovered by checking the variation of the direction among every two points of the trajectory with the minimal direction change threshold. A maximal tolerance is additionally used to verify if the point with direction change is noise or a candidate stop point. Although direction does not working weel in many cases, e.g. the trajectory of a car always has stable direction even when it stops, it can be significantly applied in some scenarios which the direction is one of the most important features such as trajectories of a fishing vessel on river.

Eventually, all of the stop discovery methods with specific features have their own advantages and disadvantages. The challenge is that such feature based method can easily fail when dealing with sparse and heterogeneous trajectory datasets. A more advanced method with hybrid or combination of features is needed.

2) Clustering-based Stop Discovery on Pure $\langle x, y, t\rangle$ data: As discussed, it is hard to choose features and combine them in computing stops. In addition, the feature like velocity, direction can be derived from the $\langle x, y, t\rangle$ data. Thus, we also propose a robust clustering based stop discovery algorithm, directly focusing on the pure $\langle x, y, t\rangle$ trajectory data. This algorithm is based on the extension of conventional DBSCAN spatial data clustering method.

DBSCAN looks for core points in order to start a cluster, later it tries to expand by adding nearby points [6]. Since we want to find personalized stops in a single trajectory respect to both space and time factors, the main concern is that stop must be a sub-trajectory of a trajectory; hence, a stop should contain only time consecutive points. We also need to overcome the problems like GPS point absence because of signal loss or low sampling rate.

Therefore, we design TrajDBSCAN following the DBSCAN principles with modification for trajectories. The method looks for core points and then expand them by aggregating other points in the neighborhood. The main differences are:

- The neighborhood is temporal linear neighborhood.
- Core point is determined based on the minimum time requirement instead of minimum number of points.

Given trajectory $\mathcal{Q}$, maximum distance threshold esp and minimum time threshold $minTime$, the aim of TrajDBSCAN is to find the sub-trajectory that all points inside is less than esp but the duration staying at this location is greater than $minTime$. Thus, in order to find personalized stops, we start with a certain point Q_k in $\mathcal{Q}$. After that, we find a sub-trajectory that is a set consecutive points $\mathcal{N}_k = \{Q_m, Q_{m+1}, ...Q_k, ...Q_n\}$ and satisfies 2 spatio-temporal constraints: (i) $\forall i, m \leq i \leq n, t_m \leq t_k \leq t_n$: $distance(Q_k, Q_i) < eps$; and (ii) $|t_n - t_m| \geq minTime$. When there exists the set $\mathcal{N}_k$, Q_k is considered as core-points and the set $\mathcal{N}_k = \{Q_m, Q_{m+1}...Q_k, ...Q_n\}$ is denoted as $esp\_linear\_neighbors$ of Q_k with respect to given parameters distance esp and time $minTime$. Each point may have a particular $\mathcal{N}_k$ and these groups possibly can overlap each other. Therefore, we use DBSCAN to merge the overlap group so as to create non-overlap stops.

Algorithm 1 describes the procedure in detail. First, we initialize an empty set of personalized stop (line 1). Then the method iterates through the trajectory and process points that have not yet been processed (line 2, 3). In line 5, the $esp\_linear\_neighbors$ of the point is computed and checked with $minTime$ constraint to exam if the point is a core-point (line 6). If a core-point is found, a new cluster is created (line 7, 8). Afterward, the method aggregate other points in the neighborhood to expand the cluster (line 9-14). Finally, the cluster is added to the personalized stop set (line 15).

Algorithm 1: TrajDBSCAN

```
input : $\mathcal{Q}$ //trajectory
        $minTime$ //minimum time
        $eps$ //neighborhood maximum distance
output: $\mathcal{PS}$ the set of personalized stops w.r.t $minTime$ and $eps$
1  $\mathcal{PS}$ = $\varnothing$
2  foreach point $Q_i$ in $\mathcal{Q}$ do
3      if $Q_i$ is unprocessed then
4          mark $Q_i$ as processed
5          $\mathcal{N}$ = Eps-Linear-Neighbors($Q_i$, $eps$)
6          if duration(C) $> minTime$ then
7              $C$ = $\varnothing$
8              $C = C \cup Q_i$
9              foreach point $Q_j$ in $N$ do
10                 if $Q_j$ is unprocessed then
11                     $C = C \cup Q_j$
12                     $\mathcal{N}'$ = Eps-Linear-Neighbors($Q_j$, $eps$)
13                     if duration($N'$) $> minTime$ then
14                         $\mathcal{N} = \mathcal{N} \cup \mathcal{N}'$
15             $\mathcal{PS} = \mathcal{PS} \cup C$
16 return $PS$
```

B. Shared Stop Discovery

This section presents two different methods in discovering shared stops. The first one is applying density method on the whole location dataset (see Fig. 5-a), and the second is based on utilizing the results of personalized stops to find the mutual locations (see Fig. 5-b).

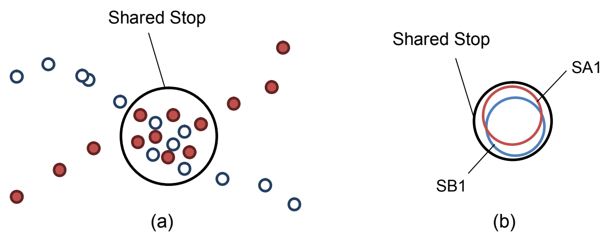


Fig. 5: Two methods to find shared stops

1) Density-based Method for Shared Stops: In this approach, the process of discovering shared stops involves applying a clustering method on the whole trajectory dataset. Several clustering algorithms have been developed in the last few years, but very few algorithms have been developed for spatio-temporal data, even less for trajectories.

Here, conventional spatial clustering methods are used frequently to find shared stops as hotspots, including DBSCAN. Different from Traj-DBSCAN, the clustering here is from multiple trajectories. There are two main parameters: a minimal number of points ($MinPts$) to build the cluster and a given distance (eps) around which the points are considered as neighbors. Such method has several advantages in clustering spatial data [6], like (1) supporting any arbitrary shape, (2) tolerant to noisy and outliers, (3) non-sensitive to the input sequence. However, it might produce fake stops as it does not consider any temporal information of trajectories.

Additional drawback when apply DBSCAN is discovering shared stops on heterogeneous trajectory dataset: considering trajectories of different users, the interruptions in data collection process and the variation of sampling rate can lead to significantly imprecise results. The more detailed analysis will be provided in the later experiment section.

2) Computing Shared Stops from Personalized Stops: As clustering method for discovering shared stops from multiple trajectories can produce fake stops, we prepose another method for computing shared stops, which is utilizing the personalized stops from Section IV-A. The rule for merging personalized stops as shared stops is following,

- Two personalized stops (A and B) are called *candidate shared stop* if the percentage of their intersection area is larger than the given threshold, i.e. $\frac{area(A) \cap area(B)}{area(A) \cup area(B)} > \delta$.
- A stop that only intersects with other stops that belonging to the same moving object will be not considered as shared stops (e.g. home of one user).

By utilizing personalized stops to discover shared stop, the approach gains two additional advantages: (i) if personalized stops discovering process considers time factor, then the shared stops process inherit this property and overcome the problem of GPS point absence or varying sampling rate; (ii) the computation cost is significantly reduced since it is based on the set of stops instead of on the whole trajectory dataset.

Algorithm 2: PersonalizedStop2SharedStop

```
input  : $\mathcal{PS}$ // a set of personalized stops
         $\delta$ //percentage threshold
output: $\mathcal{SS}$ the set of shared stops w.r.t threshold $\delta$
1  $\mathcal{SS}$ = $\varnothing$
2  foreach stop $S_i$ in $\mathcal{PS}$ do
3      if $S_i$ is unprocessed then
4          mark $S_i$ as processed
5          $N$ = getOverlapNeighbors($S_i$, $\delta$)
6          foreach stop $S_j$ in $N$ do
7              if $S_j$ is unprocessed then
8                  $N'$ = getOverlapNeighbors ($S_j$, $\delta$)
9                  $N$ = $N$ joined with $N'$

10         if $size(N) > 0$ then
11             $S$ = Convex_Hull($N$)

12         else
13             mark $S_i$ as noise

14         $\mathcal{SS}$ = $\mathcal{SS} \cup S$

15 return $\mathcal{SS}$
```

Algorithm 2 describes the detail of such method: first, it iterate the personalized stop set and compute overlapped-neighborhood for each unprocessed stop (line 3-5); second, it expand stops and aggregate one stop with other neighborhood ones (line 6-9). If there is more than one stop in the neighborhood, a shared stop is computed, by using the *convex hull* to computing a merged stop area (line 11); otherwise, the stop is marked as non-shared stop (line 13).

V. Additional Challenges in Stop Discovery

In this section, we discuss the advanced challenges in stop discovery, including both (1) building stop hierarchy (concrete stops vs. generic stops) and (2) working robustly for the heterogeneous datasets like diverse movement modes and noisy dataset with low sampling frequncy.

A. Hierarchical Stop Analysis

The aim of multi-level discretization is to generate hierarchical stops, the concrete stops at lower level are merged to produced more generic stops for higher level. At the lowest level, stop is the most concrete and known as atom-stop. Multi-level discretization have two steps: (1) generating atom-stops and (2) merging related stops.

In the first step, personalized stops are processed to create atom-stop. Personalized stops which span in large area and have long duration were split to several atom-stops based on threshold provided by the distribution information of stops, such as the average stop duration, the average stop size etc. (see Fig. 6) After additional cleansing process to remove overlapping, we eventually have a set of non-overlap atom-stops which represent the lowest level of the hierarchy. Afterwards, the second step generates higher level stops by merging related atom-stops based on distance criteria.

Given two inputs: (i) $maxheight$ - the height of hierarchical tree of stops, and (ii) $\langle d_1, d_2, ..., d_{maxHeight} \rangle$ - the set of distance to merge, where $d_1 < d_2 < ... < d_{maxHeight}$, we produce a set of stop $\mathcal{H} = \{\mathcal{S}_l | l \in [1, maxHeight]\}$. Assuming $\mathcal{S}_l = \{s_1, s_2..., s_n\}$ is the set of stops and s_i is a particular stop at level l; d_l is the distance threshold to merge two neighbor l-level stops.

Algorithm 3 describes the details to produce stops S_{l+1} of level $l+1$ from stops S_l of level l. First, we iterate stops of level l. For each unprocessed stop, the distance-neighborhood is computed (line 2-4). After that, the method aggregates other stops in the neighborhood and expand (line 5-7). Then a level l stop is computed using the convex hull as stop area (line 8).

Algorithm 3: HierarchyDiscover

```
input  : $\mathcal{S}_l = \{s_1, s_2, ..., s_n\}$ // shared stop at level $l$
         $d_l$ // distance parameter
output: $\mathcal{S}_{l+1}$ shared stop at level $l+1$
1  $\mathcal{S}_{l+1}$ = $\varnothing$
2  forall $s_i$ in $\mathcal{S}_l$ do
3      if $s_i$ is unprocessed then
4          mark $s_i$ as processed
5          $N$ = getNeighbors($s_i$, distance)
6          foreach stop $S_j$ in $N$ do
7              if $s_j$ is unprocessed then
8                  $N'$ = getNeighbors($S_j$, distance)
9                  $N$ = $N$ joined with $N'$

10         create stop $S$ = Convex_Hull($N$)
11         add stop $S$ to $\mathcal{S}_{l+1}$

12 return $\mathcal{S}_{l+1}$
```

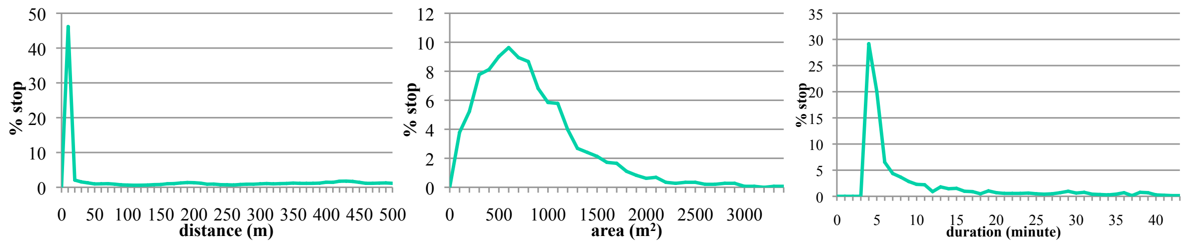


Fig. 6: Stop distribution information (diameter distance, area size, time duration of stops)

B. *Discovering stops on sparse and diverse dataset*

1) Sparse Dataset: The concern of sparse dataset is the low sampling rate or even big time gap between two consecutive records. As a result, the features like *velocity*, *direction* calculated by two neighboring points become meaningless. Existing methods usually fail in identifying stops in such cases.

For example, Fig. 7 (a) shows two consecutive points A and B. The actual path is the dashed line $|\widetilde{AB}|$ which is much longer than the solid line $|AB|$, which can bring fake stop when using velocity $v = \frac{|AB|}{T}$. Another case Fig. 7 (b), a person staying in a building, but only two points collected (A and B), respectively pointing to his entering and leaving the building, as there is no signal inside the building. With traditional density method, we cannot find such stop.

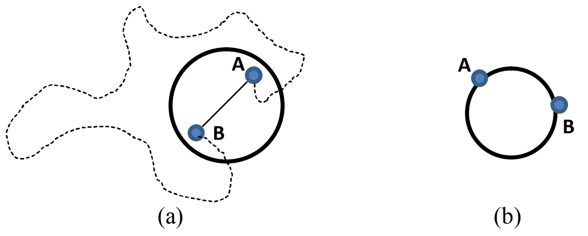


Fig. 7: False positive & false negative of stops in sparse data

In order to deal with such sparse datasets, our TrajDBSCAN considers both the spatial density and the time information. By adding $minTime$ (the minimum stop time duration), TrajDBSCAN can handle the big temporal gap and find stops.

2) Diverse Trajectory Datasets: Trajectories can have two kind of diversity: (1) sampling frequency is dynamically tunable due to the energy issue, i.e. when moving slowly it can use low sampling rate whilst moving quickly it takes high sampling rate; (2) the transportation modes of movement can also be diverse, e.g. people can take metro, bus, cycling, and even walking etc. Traditional methods based on features or density could not work well with such diverse datasets. Therefore, we need a robust algorithm in discovering stops in different kinds of non-stationary trajectory datasets.

The experiment section will discuss the details and show the tolerance of our TrajDBSCAN algorithm which can work well in different kinds of trajectory datasets, such as vehicle trajectories as well as people trajectories from smartphones which have much more diversity.

VI. Experiment

We validate our experiment within two big datasets, one of car trajectories and the other of people daily movement.

- Nokia Dataset: The dataset contains 178667 GPS points of 6 users moving around Lausanne, Switzerland in one year, from Nokia Research Center in Lausanne. Raw GPS data of each user was segmented into trajectories by day. After segmentation and preprocessing steps to remove short duration trajectories, there are 2324 trajectories.
- Milan Dataset: It has 190779 GPS points of 4162 private cars moving around Milan, Italy in one week. The data was collected in the GeoPKDD project. We have 5749 trajectories after segmenting by day and preprocessing to eliminate short duration trajectories.

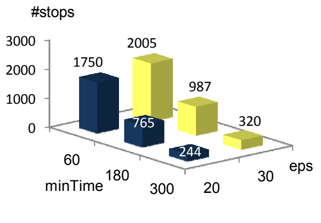
The Nokia GPS data is collected every 10 seconds. Its sampling rate is quite fine in compare with that of Milan dataset, whose collecting time is varied from 10 seconds to 600 seconds. However, smartphone user of Nokia dataset employed varying transportation modes like walking, cycling, metro, bus etc., which make the dataset more diverse and sparse.

As mentioned, we focus on dynamical stop discovery without using predefined geographic information; our main experiments do not deal with ROIs and POIs. However, we will use them to properly analyze and validate the soundness of the stops which was discovered.

Our experiment is organized as follows: First we exhibit TrajDBSCAN to discover personalized stops and compare it with other methods; second, we carry out the experiment in discovering shared stops; third, we test our method on building stop hierarchy; and finally, we show experimental results that prove the soundness of our TrajDBSCAN algorithm when applied on sparse and diverse trajectory datasets.

A. *Personalized Stop Discovery*

1) TrajDBSCAN: In order to get the best results in finding stops, we tune and analyze the sensitivity of two important parameters, i.e. $minTime$, eps. Fig. 8-(a) indicates the number of stops discovered by different parameter values. It workths noting that the number of stops drops dramatically as $minTime$ increases; when $minTime > 300s$, there are very few trajectories with more than one stop. However, if $minTime$ is too low there are many fake stops which can be traffic jams, congestions, crossroads, etc. Thus, we can indicate that 180 second is the most appropriate value for $minTime$.



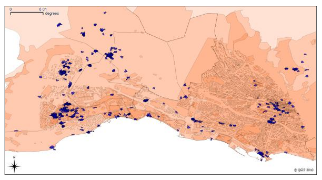


Fig. 8: (a) sensitivity analysis, (b) stops of Nokia data

We can evaluate our results with real POI data that extracted from Openstreetmap. Setting eps as 20m seems to be too strict as it makes TrajDBSCAN lost some stops when compared with $eps = 30m$. Therefore, 30m is likely a better value for eps in discovering stops in Nokia data.

To avoid both fake stops and missing stops, we use $minTime = 180s$ and maximum neighborhood distance $eps = 30m$ for the final experiment. Fig. 8-(b) illustrates all stops discovered by TrajDBSCAN with such parameters. We can observe most of the stops are in EPFL campus, student residential areas, and Lausanne city center. In general, this result make much sense as most of users of Nokia tracking dataset are EPFL students.

2) Comparing TrajDBSCAN with other approachs: This section compares our TrajDBSCAN methods with other velocity-based or density methods.

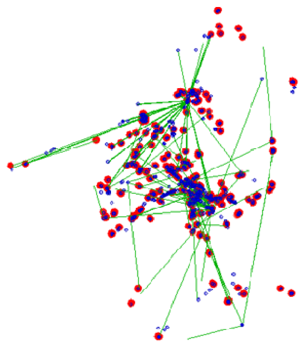


Fig. 9: Stops discovered by velocity vs. TrajDBSCAN

Fig. 9 shows the different between our approach with the velocity approach in [18] of a Bus trajectory dataset from RTreePortal[3]: the red dots are stops TrajDBSCAN computed whilst the blue ones are from velocity metrics. We can observer that our method can avoid much more faked stops.

In addition, we compare TrajDBSCAN with other density-based method, e.g. CB-SMoT [14] – a clustering method based on the speed variation of trajectories. The parameters of CB-SMoT in our experiment are using speed limit as 1.5m/s, maximum average speed as 1.2m/s, minimum time as 180s, which is suggested from their paper.

To further compare different methods in discovering stops, we provide a metric called "Stop-based Trajectory Similarity", which is determined by the longest common stop sequence.

[3]http://www.rtreeportal.org/

Definition 3 (Stop-based Trajectory Similarity)**.** *Given trajectory $\mathcal{Q}_a$ and trajectory $\mathcal{Q}_b$ that abstracted by the sequence of stops: $\mathcal{Q}_a = \{S^a_1, S^a_2, ..., S^a_n\}$; $\mathcal{Q}_b = \{S^b_1, S^b_2, ..., S^b_m\}$. If there exists k common stops, which are also in the same sequence in both $\mathcal{Q}_a$ and $\mathcal{Q}_b$, then the stop-based trajectory similarity measure between $\mathcal{Q}_a$ and $\mathcal{Q}_b$ is : $sim(\mathcal{Q}_a, \mathcal{Q}_b) = \frac{2*k}{m+n}$.*

The determine of common stops can use the previous measurement of $\frac{area(A) \cap area(B)}{area(A) \cup area(B)} > \delta$.

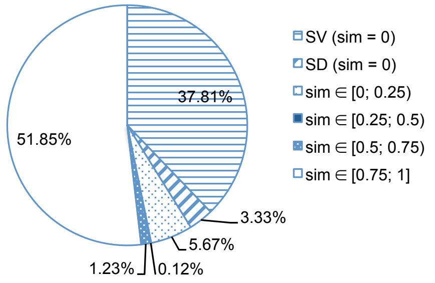


Fig. 10: The distribution of sim (CB-SMoT vs. TrajDBSCAN)

Taking Nokia dataset for example, we choose 812 trajectories which have at least one stop computed by either CB-SMoT or TrajDBSCAN. We calculate sim for each trajectory and check the distribution of these 812 daily trajectories (see Fig. 10). In the figure, there are six categories, i.e. $sim \in [0, 0.25], (0.25, 0.5], (0.5, 0.75], (0.75, 1]$, with two special case when $sim = 0$, one is SV (the percentage of trajectories only have stops by CB-SMoT.) and the other is SD (the percentage of trajectories only have stops by TrajDBSCAN.). It worths noting two main observations: (1) most of the trajectories have high similarity degree (above 0.75) between the two methods; (2) there is a high percentage of SV (say about 37.8%), which means CB-SMoT produces more small or even fake stops.

Another interesting analysis is the shape of stops. Based on the previous parameters in CB-SMoT, we observe the large spanning stop area (see Fig. 11-a), which is obviously incorrect. In order to avoid such long cluster, we decrease the maximum average speed to 1.0m/s. However, as shown in Fig 11-b, the result is even worse that making two clusters; we only obtain cluster on the road and almost nothing from the real stop. The truth is that the moving object (mobile phone user) walks with a very low speed on the road. On the contrary, our TrajDBSCAN can discover such stop more precisely, as shown in Fig 11-c.

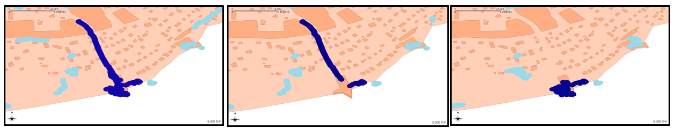


Fig. 11: Personalized stops discovered by different methods

In summary, for the stationary dataset like small dataset from RTreePortal, the results are quite similar between our method and others (see Fig. 9); whilst for more diverse and sparse dataset, TrajDBSCAN can produce much better results than existing ones (see Fig. 11). This is because our approach can eliminate fake stops, and discover stops more robustly.

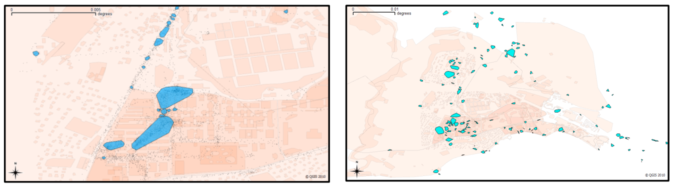
B. Shared Stop Discovering

This part validates the two methods for finding shared stops, i.e. the density-based method on the whole dataset and the refinement of "personalized" stops to compute shared stops.

1) Density-based Clustering on the whole dataset: Fig 12-a shows the shared stops discovered when applying DBSCAN on the complete Nokia trajectory dataset. There are 129 stops discovered. After mapping these shared stops to the POIs in Lausanne area and comparing them to personalized stops we computed before, we learned that many stops are missing. Therefore, we relax the density requirement of DBSCAN in order to discover more stops. However, this in turn bring many fake stops that are lying at the middle of roads. Such results prove our hypothesis that such density-based stop clustering on the whole trajectory dataset bring a large amount of fake stops as many trajectories pass by each other and make a lot of short-term overlaps. Such overlaps are not real stops.

2) Shared stops from "personalized" stops: To reduce the number of fake shared stops and utilize the personalized stops that computed by TrajDBSCAN method, we extract shared stops from the "personalized stops" computed from each single trajectory separately. After refining 987 personalized stops, 524 shared stops are discovered. To further ensure the accuracy of these shared stops, we compare them with the POIs in Lausanne area. After the matching process, nearly 80% of the shared stops can match to at least one POI.

In addition, Fig. 12 intuitively sketches the results of such shared stops discovery between the two methods, we can further observe better accuracy from the second method.



(a) Density-based shared stops (b) Shared stops from personalized stops

Fig. 12: Shared stops discovered by two methods

C. Build Stop Hierarchy

The objective of this experiment is to validate the results of building stop hierarchy, which is based on using the results of the personalized stops discovered by TrajDBSCAN in the early experiment.

Firstly, we refine personalized stops into non-overlapping atom-stops. To determine the suitable size of atom-stops, we analyze the distribution information of personalized stops. As shown in Fig. 6, we can observe the stop distribution in terms of different features, i.e. the diameter distance, the area size, and the time duration respectively. According to the stop distribution, we can provide suitable threshold (e.g. $\delta \times mean$, where the $mean$ can be the average value of stop distance, area, or duration.) to determine whether stops need to be split or merged – building the stop hierarchy.

Fig. 13 illustrates the result of computing stop hierarchy, where we build 4 levels of stops, from the atomic level (L0) to the highest level (L4). The subfigure 13-a shows the number of stops at each individual level. As we can see, there are 114 single-child stops at level-3. These stops are very far from other stops, therefore they cannot be merged with others to form higher stops though the merging-distance of the level is 250m. On the contrary, there are 84 stops having at least one neighboring stop in radius less than 30m. Similar to the previous POI matching process for validation, we check the ROI data from Openstreetmap. The interesting observation is that most of the multi-child stops are the importance places in Lausanne such as: Flon (city center), two universities (EPFL and UNIL), Ouchy (lake port), Renens Center etc. These areas are exactly hotspots in the trajectory dataset area. To provide a visual hierarchy of such hotspots (stops), Fig. 13-b visualizes the hierarchy of stops around EPFL. We can observe: the first level includes the buildings inside EPFL, the second level is EPFL campus, and the third level is a big area contains EPFL, UNIL, and sport center at the lakeside.

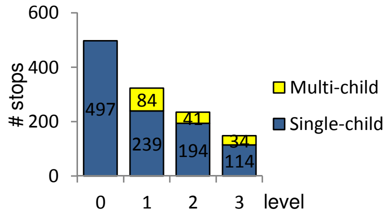
D. Sparse and Diverse Trajectory Dataset

The last experiment part is to validate the TrajDBSCAN method for discovering stops in sparse and diverse trajectory dataset. Such heterogeneous trajectories can bring much high errors in discovering stops, such as generating fake stops (false positive) or missing real stops (false negative).

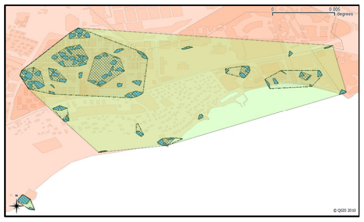
In the area with GPS signal (e.g. outside buildings), Nokia dataset has very high sampling frequency 1/10, which means GPS data is recorded once in 10 seconds. To test our method in different sampling rate, we modify the dataset to get different sampling rate by removing the data points – manually generating more sparse dataset (e.g. from 1/10 to 1/220 in Fig. 14). The figure shows the results of our TrajDBSCAN comparing with the CB-SMoT method regarding dataset with different sampling rates.

In Fig. 14, TrajDBSCAN has a very stable result when the sampling rate is between 1/10 and 1/160, where the stop number only has a slight decrease because some short trajectories do not hold enough GPS points after removing some for simulating dataset with low sampling frequency. When sampling rate further reduced to 1/190 and 1/220, the number of stops decreases significantly. This is because $minTime$ that we used is 180 seconds, which is already less than the time gap between two consecutive GPS points.

Obviously in Fig. 14, the stop number computed by CB-SMoT is much sensitive to the sampling rate. The more spare data, the stop number drops more significantly. Therefore, TrajDBSCAN is much more robust with different sampling frequency as long as the sampling rate is covered in the range of minimum time parameter.



(a)



(b)

Fig. 13: Results of Hierarchical Stop Discovery

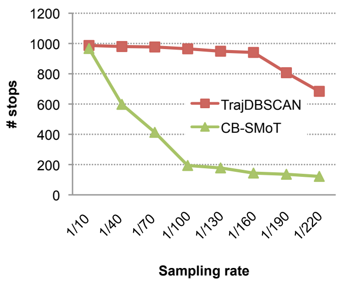


Fig. 14: #stops w.r.t. sampling rate

Our final experiment validates TrajDBSCAN in another dataset. It is about 4162 private cars moving around Milan, Italy in one week. After data preprocessing, Milan dataset contains 2283 trajectories and has a very unstable sampling rate, varying in a wide range from 1/10 to 1/600.

As Fig. 14 has already shown that TrajDBSCAN could fail to discover stops when minimum time parameter does not cover the sampling frequency range, we use $minTime$ no less than 600s. Fig. 15 illustrates the number of stops when applying TrajDBSCAN on the Milan dataset w.r.t. different values of $minTime$ and eps. From Fig. 15, we can observe the number of stops is quite stable.

In a short summary, our method works more robustly for sparse and diverse trajectory data which might have very unstable sampling frequency, or GPS data from smartphones that has movement with different transportation modes.

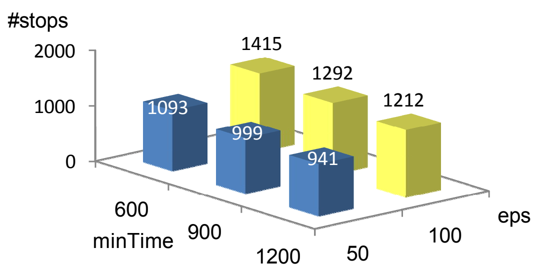


Fig. 15: #stops by TrajDBSCAN on Milan dataset

VII. Conclusion

In this paper, we provide a robust algorithm on stop discovery, which can work well in heterogeneous trajectories where (1) GPS sampling frequency can be different and irregular, (2) GPS feeds are sparse with big gaps, (3) moving objects can be vehicles with stationary speeds or peoples with non-stationary movement velocity as people can take different transportation modes like car, metro, walking etc. In addition, our approach can (1) find both *personalized stops* and *shared stops* with several methods; and (2) build the *stop hierarchy* either from pure GPS records by clustering or based on previously discovered personalized stops; the stop hierarchy can represent stops in different granularity at different levels.

Our ongoing and future work is focusing on semantic stop analysis, and further inferring movement behaviors from the computed stops at different perspectives.

References

[1] G. Agamennoni, J. Nieto, and E. M. Nebot. Mining GPS Data for Extracting Significant Places. In *ICRA*, pages 855–862, 2009.

[2] L. O. Alvares, V. Bogorny, B. Kuijpers, J. Macedo, B. Moelans, and A. Vaisman. A Model for Enriching Trajectories with Semantic Geographical Information. In *ACM-GIS*, page 22, 2007.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *SIGMOD*, pages 49–60, 1999.

[4] D. Ashbrook and T. Starner. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

[5] X. Cao, G. Cong, and C. S. Jensen. Mining Significant Semantic Locations From GPS Data. *PVLDB*, 3(1):1009–1020, 2010.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pages 226–231, 1996.

[7] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory Pattern Mining. In *KDD*, pages 330–339, 2007.

[8] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of Convoys in Trajectory Databases. In *VLDB*, pages 1068–1080, 2008.

[9] N. Kami, N. Enomoto, T. Baba, and T. Yoshikawa. Algorithm for Detecting Significant Locations from Raw GPS Data. In *Discovery Science*, pages 221–235, 2010.

[10] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering. In *VLDB*, pages 1081–1094, 2008.

[11] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory Clustering: a Partition-and-Group Framework. In *SIGMOD*, pages 593–604, 2007.

[12] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal Outlier Detection in Vehicle Traffic Data. In *ICDE*, pages 1319–1322, 2009.

[13] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[14] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *SAC*, pages 863–868, 2008.

[15] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot. A Conceptual View on Trajectories. *Data and Knowledge Engineering*, 65:126–146, 2008.

[16] K. Xie, K. Deng, and X. Zhou. From Trajectories to Activities: a Spatio-Temporal Join Approach. In *GIS-LBSN*, pages 25–32, 2009.

[17] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and A. Karl. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *EDBT*, 2011.

[18] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty. A Hybrid Model and Computing Platform for Spatio-Semantic Trajectories. In *ESWC*, pages 60–75, 2010.

[19] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining Interesting Locations and Travel Sequences from Gps Trajectories. In *WWW*, pages 791–800, 2009.

[20] C. Zhou, D. Frankowski, P. J. Ludford, S. Shekhar, and L. G. Terveen. Discovering Personally Meaningful Places: an Interactive Clustering Approach. *ACM Trans. Inf. Syst.*, 25(3):12, 2007.

[21] M. Zimmermann, T. Kirste, and M. Spiliopoulou. Finding Stops in Error-Prone Trajectories of Moving Objects with Time-Based Clustering. In *Intelligent Interactive Assistance and Mobile Multimedia Computing*, pages 275–286. 2009.