

# Online Clustering of Trajectories in Road Networks

Ticiania L. Coelho da Silva  
Insight Data Science Lab, Brazil  
ticianalc@insightlab.ufc.br

Francesco Lettich  
University of Alberta, Canada  
francesco.lettich@gmail.com

José Antônio Fernandes de Macêdo  
Insight Data Science Lab, Brazil  
jose.macedo@insightlab.ufc.br

Karine Zeitouni  
DAVID Lab, UVSQ - Université Paris-Saclay, France  
karine.zeitouni@uvsq.fr

Marco A. Casanova  
Department of Informatics - PUC-Rio, Brazil  
casanova@inf.puc-rio.br

**Abstract**—The ubiquity of GPS-enabled smartphones and automotive navigation systems allows to monitor and collect massive streams of trajectory data in real-time. This enables real-time analyses on mobility data in urban settings, which in turn have the potential to substantially improve traffic conditions, analyze congested areas, detect events in (quasi) real-time, and so on. While many existing approaches characterize past movements of moving objects from historical trajectory data, or address the problem of finding out clusters of moving objects from data streams, such approaches fail to capture how movement behaviors unravel over time – for instance, they fail to capture typically trafficked routes or traffic jams. In this work we propose NET-CUTiS, a novel approach that addresses the problem of discovering and monitor the evolution of clusters of trajectories over road networks from trajectory data streams. We conduct several experiments that demonstrate the validity of our proposal in terms of clustering quality and run-time performance.

**Index Terms**—trajectory, clustering, road network

## I. INTRODUCTION

The ubiquity of GPS-enabled smartphones and automotive navigation systems connected to the Internet allows to monitor and collect massive streams of trajectory data in real-time. This enables various kinds of real-time analyses on mobility data – for instance, real-time detection of regularities (e.g., typical traffic flows over a road network) and anomalies (e.g., traffic jams) – thus allowing institutions to quickly act when facing urgent decision-making tasks in urban settings. Indeed, the ability to capture and monitor the evolution of *clusters* (i.e., groups) of moving objects that exhibit *similar movement behaviors* allows to gain valuable insights on various kinds of mobility patterns [1]–[3]. For example, a traffic jam can be seen as a sequence of merges of distinct clusters of objects, or as an evolution of clusters of objects which exhibit similar movements in space and time. Also, tracking small changes in the behaviors of moving objects allows the possibility to detect in (quasi) real-time traffic jams, and to predict their duration.

In this paper we consider trajectory data streams and address the problem of detecting clusters of moving objects trajectories constrained by road networks while keeping track of the evolution of previously detected clusters over time. This problem poses multiple challenges due to the potential volume of data and events to process. Indeed, moving objects typically update their position frequently and continuously. Previously unseen

moving objects may appear at any time instant, while others may disappear. Strategies that can be used in combination to tackle the above challenges are (1) the adoption of time *discretization*, as this facilitates the processing of streams, and (2) the use of *incremental* clustering, i.e., the use of data structures that allow to represent clusters and their evolution over time (which in turn avoids re-discovering previously detected clusters).

In the literature there exist several approaches that perform (incremental) clustering of moving objects positions, and monitor the evolution of clusters (or patterns, in general) over time. Among these, the most notable ones are [3]–[7]. We report that such approaches limit themselves to Euclidean spaces and instantaneous positions, and thus fail to capture mobility behaviors that unravel over time, e.g., they fail to capture typically trafficked routes or traffic jams. On the other hand, other approaches [1], [2], [8], [9] consider moving objects trajectories constrained by road-networks. Even though these solutions adopt time discretization (i.e., time is partitioned in intervals to reduce computational costs), they do not employ incremental clustering as they recompute clusters from scratch at every time interval. This, in turn, leads to very high computational costs when dealing with big volumes of data.

In this paper we address the above problem by providing a novel on-line incremental clustering algorithm, NET-CUTiS, that processes data streams of moving objects trajectories constrained by a road-network. NET-CUTiS leverages a novel spatio-temporal distance function to determine the *proximity* between two trajectories constrained by a road network. The extensive experimental evaluation conducted over a real-world dataset assesses and shows the validity of NET-CUTiS in terms of *clustering quality* and *run-time performance*. We argue that our study is relevant as, differently than existing literature, we propose an incremental on-line approach that operates on *road networks* (rather than Euclidean spaces) and is able not only to discover clusters but, more crucially, to monitor their *evolution* over time, and in doing so NET-CUTiS exploits both *spatial* and *temporal* information.

The remainder of the paper is structured as follows: Section II provides some preliminary notions and the problem statement. Section III presents NET-CUTiS. Section IV discusses the related works. Section V presents the experimental eval-

uation. Finally, Section VI draws the final conclusions.

## II. PRELIMINARIES AND PROBLEM STATEMENT

We assume that a road network is modeled by a directed graph  $G(V, E, W)$ , where the vertices in  $V$  represent road intersections (or points of interest), the edges in  $E$  represent road segments, while  $W$  represents the weights associated with the edges in  $E$ . In this setting we assume that the weight of an edge connecting two vertices  $v_i$  and  $v_j$ , and we denote it by  $w(v_i, v_j) \in W$ , is given by the *length* of the associated road segment.

In this work we assume that spatial coordinates of moving objects are map-matched to the edges of the road network via suitable algorithms – we refer to [10] for an exhaustive overview of the state-of-the-art. We thus conveniently define the trajectory  $TR_j = \langle p_1, \dots, p_n \rangle$  of a moving object  $o_j$  as a temporally ordered sequence of edges  $o_j$  crossed during some time interval. We represent each element  $p_i \in TR_j$  as a pair  $p_i = (e_i, [t_i, t_w])$ , where  $e_i \in E$  represents the edge that  $o_j$  crossed during the time interval  $[t_i, t_w]$ . We also conveniently introduce the shorthand  $TR_j([t_i, t_w]) = e_i$  to indicate the edge  $e_i \in E$  that  $o_j$  crossed during the time interval  $[t_i, t_w]$ . Finally, we define a *sub-trajectory*  $ST_j$  of  $o_j$  to be a *subsequence* of consecutive edges that can be extracted from  $TR_j$ . We can now introduce the notion of *trajectory stream* of a *time window*, which represents the foundation of the time discretization strategy we use to process streams of trajectories.

**Definition 2.1 (Trajectory stream of a time window):** Let  $i = [t, t + \Delta t)$  be some *time window* (i.e., temporal interval). We define the *trajectory stream*  $I_i = \{ST_1^i, ST_2^i, \dots, ST_n^i\}$  to be the set of sub-trajectories whose movements are restricted to the time window  $i$ . As such,  $ST_j^i$  represents the restriction of the trajectory  $TR_j$  of the object  $o_j$  to the time window  $i$ .

The use of time discretization requires to appropriately manage certain events that may occur while tracking moving objects, as these directly influence the detection and monitoring of clusters (and thus the correctness of such operations). For instance, (1) a new moving object may *appear* in the system, hence such object must be inserted into the system. On the other hand, (2) a moving object may *disappear* from the system, hence such object must be removed from the system. Finally, (3) moving objects mainly *update* their position over time, and thus the system should be able to track them all along their evolution during successive *time windows*. Overall, these events have the potential to influence the *creation* of new clusters or *alter* the state of existing ones (and thus influence their evolution). Managing event (1) is straightforward. For what concerns event (2), we ensure that when a moving object  $o_j$  stops sending its updates during some time window  $i$ ,  $o_j$  is discarded when the time window  $i + 1$  starts.

Our main goal is to track moving objects and incrementally discover, as well as monitor the evolution of, sub-trajectory clusters in a road network. We thus state the problem we intend to address as follows.

**Problem Statement.** Given a road network  $G = (V, E)$  and a trajectory stream  $I_i = \{ST_1^i, ST_2^i, \dots, ST_n^i\}$  associated with a time window  $i$ , we aim at (1) *detecting* clusters of sub-trajectories in  $G$  that exhibit similar movement behaviors and (2) *keep track* of the *evolution* of clusters of sub-trajectories that were already found within time windows *preceding*  $i$  – this implies that previously detected clusters should not have to be re-discovered from scratch at each time window. Finally, clusters should be discovered or monitored by taking advantage of movement similarities between objects, and in doing so both spatial and temporal information should be considered.

## III. NET-CUTiS

In this section we present NET-CUTiS, the approach we propose to target the problem we are considering. The section first introduces the spatio-temporal distance function used to measure the proximity between trajectories constrained by a road network. Subsequently, the section proceeds to present the approach.

### A. Spatio-temporal proximity between trajectories

The spatio-temporal distance function proposed in this paper leverages the notion of *network path* [11]. Network paths represent paths within a road network expected to represent major traffic flows and are extracted from historical trajectory data. Roughly speaking, the extraction process starts by finding out the edge traversed by the largest number of trajectories and then expands it to form a path shared by some historical trajectories. The edges associated with such path are then removed from the network, and the above procedure is iterated over the remaining edges until no trajectory remains to be considered. In our work NET-CUTiS uses network paths, as well as a trajectory compression algorithm, to efficiently find out clusters of objects exhibiting similar movement behaviors.

Once a set of network paths becomes available, NET-CUTiS uses them to compress sub-trajectories within a trajectory stream. This allows to make comparisons between trajectories at a coarser granularity, which in turn requires less computational efforts. To this end we adapt the compressed representation proposed in [11] to the context of data mining. Indeed, differently from the context of trajectory queries, which requires to precisely know the position of trajectories within the road network, having the exact starting and ending positions of trajectories within an edge is not so important when clustering them. Therefore, we adapt the definitions provided in [11] by dropping exact positions of trajectories within the edges they traverse. We also define a compressed trajectory to be a sequence of network paths, where each network path in a sequence is coupled with a temporal interval representing the period in which the trajectory traversed that path. We follow the same procedure provided in [11] to partition the road network (as already illustrated above) and use the resulting network paths to compress (and thus represent) trajectories within a stream. Let us now introduce the notion of *sub-trajectory compression*.

**Definition 3.1 (Sub-trajectory compression):** Let us consider a sub-trajectory  $ST_j^i = \langle (e_1, [t_1, t_2]), (e_2, [t_2, t_3]), \dots, (e_n, [t_n, t_{n+1}]) \rangle$ . Let us also consider a set of network paths  $NP = \{np_1, \dots, np_m\}$  that were previously derived from historical data. We then compress  $ST_j^i$  via  $NP$  by means of the algorithm proposed in [11]. Specifically: let  $Edges_{ST_j^i} = \{e_1, e_2, \dots, e_n\}$  be the sequence of edges associated with  $ST_j^i$ . Then, given a *compression threshold*  $\gamma$  we claim that a network path  $np \in NP$  can be used to represent a sub-path of  $ST_j^i$  if the following condition holds:

$$\frac{|Edges_{ST_j^i} \oplus np|}{|np|} \geq \gamma,$$

with  $Edges_{ST_j^i} \oplus np$  representing the longest common *subpath* between  $np$  and  $Edges_{ST_j^i}$ .

Given the above definition, from now on we represent a compressed sub-trajectory  $ST_j^i$  as a sequence of network paths, and we express this by means of a sequence of pairs  $ST_j^i = \langle (np_1, [t_{init}^1, t_{end}^1]), \dots, (np_r, [t_{init}^r, t_{end}^r]) \rangle$ , where  $np_k \in NP$  represents the  $k$ -th network path that the moving object  $o_j$  traversed during the time interval  $[t_{init}^k, t_{end}^k]$ . As already argued before, in this work our goal is to cluster trajectories and in this context giving them a coarse representation suffices. As such, when compressing a trajectory we leave out those sub-paths that do not match any network path. Note that sub-trajectory compression preserves temporal information. In general, choosing a proper compression threshold  $\gamma$  translates into finding an appropriate degree of compression. At this point we leverage the notion of network paths and introduce the spatio-temporal distance function we use to determine the *proximity* between pairs of sub-trajectories.

**Definition 3.2 (Spatio-Temporal Distance Function):**

Let  $i = [t, t + \Delta t]$  be a time window, and let  $ST_j^i$  and  $ST_k^i$  be two sub-trajectories we desire to compare in terms of spatio-temporal proximity. Let  $n = \min(|ST_j^i|, |ST_k^i|)$  be the minimum number of network paths between  $ST_j^i$  and  $ST_k^i$ . Let then  $match_r(ST_j^i, ST_k^i, \delta)$  be a function that returns 1 if it exists a time interval  $p \subseteq [\max(t_{init}^r - \delta, t), \min(t_{end}^r + \delta, t + \Delta)]$  such that the  $r$ -th network path traversed by  $ST_j^i$  during the time interval  $[t_{init}^r, t_{end}^r]$  coincides with the one traversed by  $ST_k^i$  during  $p$ , and 0 otherwise. Then, we define the *spatio-temporal distance* between  $ST_j^i$  and  $ST_k^i$  as follows:

$$distance(ST_j^i, ST_k^i, \delta) = \begin{cases} 1 - \frac{\sum_{r=1}^{n=|ST_j^i|} match_r(ST_j^i, ST_k^i, \delta)}{\max(|ST_j^i|, |ST_k^i|)} & \text{if } |ST_j^i| \leq |ST_k^i| \\ 1 - \frac{\sum_{r=1}^{n=|ST_k^i|} match_r(ST_k^i, ST_j^i, \delta)}{\max(|ST_j^i|, |ST_k^i|)} & \text{otherwise,} \end{cases} \quad (1)$$

Intuitively, the fraction past the minus operation measures the similarity between two trajectories – the closer the value of *distance* to zero, the greater their spatio-temporal proximity.

The use of network paths within a distance function allows to reduce computational costs than other functions available in the literature, with the most notable one (and applicable to NET-CUTiS) being the Synchronous distance. If  $n_1$  and  $n_2$  denote the number of position updates of two trajectories, then [12] shows that the Synchronous distance requires to perform  $O(n_1 + n_2)$  distance comparisons. The use of network paths greatly reduces such computational cost. We concede, however, this benefit can possibly impact accuracy negatively.

### B. Incremental Trajectory Clustering Algorithm

In this section we introduce NET-CUTiS, an online incremental trajectory clustering approach that operates on streams of trajectory data constrained by road networks to discover and keep track of the evolution of clusters of trajectories over time. NET-CUTiS is structured into two distinct *components*: the *first* one constitutes an independent pre-processing step which goal is to find out a set of network paths from carefully selected historical trajectory data. The *second* component is a pipeline in charge of processing trajectory data streams. Such pipeline operates by discretizing the time in time windows (intervals), each having size  $\Delta$ : at the end of a time window  $i$ , the pipeline considers the associated trajectory stream  $I_i$  and serially executes four distinct phases to discover new clusters or keep track of the evolution of previously found ones. In the following we provide the details behind the two components.

1) *Extraction of network paths*: NET-CUTiS first requires to pick up carefully selected historical trajectory data to extract network paths that are expected to be representative of traffic flows within the considered geographical area. Once such data is available, NET-CUTiS employs the algorithm introduced in [11] (provided with a compression threshold  $\gamma$ ) to extract a set of network paths we denote by  $NP$ . Note that this step can be executed independently from the pipeline in charge of processing a trajectory data stream, as long as a set of network paths is available to the latter.

2) *Pipeline, phase I – sub-trajectory compression*: once some set of network paths  $NP$  is available, NET-CUTiS can employ the pipeline in charge of processing a trajectory data stream. During the pipeline's *first phase* the goal is to *compress sub-trajectories* available in a trajectory stream  $I_i$  by means of  $NP$ . To do so, NET-CUTiS applies the criterion outlined in Equation 3.1 to output the set of compressed sub-trajectories. In the phases that follow only sub-trajectories that were compressed are used for cluster detection and maintenance, while incompressible ones are considered as *outliers* – indeed, we assume that the latter do not fall within representative traffic flows and are thus ignored accordingly.

3) *Pipeline, phase II – selection of representative trajectories*: the pipeline's second phase aims at determining the set of *representative trajectories* associated with a trajectory stream  $I_i$ . Intuitively, a sub-trajectory of  $I_i$  is deemed to be representative if it *best represents* the movement behaviors of a specific group of sub-trajectories of  $I_i$ . Let us denote by  $ST_j^i$  the sub-trajectory representing the movements of a moving object  $o_j$  limited to the time window  $i$ . To determine

if  $ST_j^i$  is representative two criteria are used: the first one assesses the *representativeness* of  $ST_j^i$  by means of a Gaussian kernel *voting* function based on the one introduced in [13]. The second one verifies if the *number* of sub-trajectories exhibiting movement behaviors similar to those of  $ST_j^i$  is above a given threshold, and to do so it leverages the notion of *neighborhood density*  $N_\epsilon$ . In the following we formalize such criteria.

**Definition 3.3 (Representative trajectory):** Let  $I_i = \{ST_1^i, ST_2^i, \dots, ST_n^i\}$  be the trajectory stream associated with a time window  $i$ . Let also  $\rho$  be the *representativeness* threshold,  $\delta$  the time tolerance,  $\sigma$  the chosen *standard deviation*,  $\epsilon$  the *distance* threshold, and  $\tau$  the *density* threshold. Then, we define  $ST_j^i$  to be a *representative* trajectory of some group of sub-trajectories  $I'_i \subseteq I_i$  if and only if the following conditions hold:

- 1)  $\exists ST_j^i \in I'_i, \forall ST_k^i \in I'_i, k \neq j,$   

$$voting(ST_j^i, ST_k^i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{distance^2(ST_j^i, ST_k^i)}{2\sigma^2}} > \rho$$
- 2) if  $N_\epsilon(ST_j^i) = \{ST_k^i \in I_i \mid distance(ST_j^i, ST_k^i, \delta) \leq \epsilon\},$   
then  $|N_\epsilon(ST_j^i)| \geq \tau$

Observe from the first condition that choosing a proper value for  $\rho$  is important: in this work we deemed reasonable to set  $\rho$  to the maximum distance *allowed* between any sub-trajectory and its representative trajectory. Such distance should be at least equal to  $\epsilon$  to keep the *neighborhood density*  $N_\epsilon$  around representative trajectories *significant*. It is finally useful to spend some words on the relationship between *distance* and *voting*: specifically, the closer *distance* to zero, the larger the result returned by *voting*. Intuitively, this implies that if  $ST_j^i$  is very close in time and space to  $ST_k^i$ , then  $ST_j^i$  is likely to be representative of  $ST_k^i$ .

4) *Pipeline, phase III – creation and maintenance of micro-groups*: the goal of this phase is to create and maintain micro-groups by means of the set of representative trajectories returned by the second phase. In this context we define a *micro-group* to be a small and dense group of moving objects that exhibit similar movement behaviors in space and time (e.g., a group of people sharing the same vehicle or a group of vehicles stuck in a traffic jam). NET-CUTiS determines the existence of micro-groups according to the definition that follows.

**Definition 3.4 (Micro-group):** Let  $I_i = \{ST_1^i, ST_2^i, \dots, ST_n^i\}$  be the trajectory stream associated with a time window  $i$ . Let also  $O_i$  be the set of moving objects associated with the sub-trajectories in  $I_i$ ,  $\epsilon$  the distance threshold,  $\delta$  the time tolerance,  $\tau$  the minimum density threshold, and  $\rho$  the representativeness threshold. Then, we define a *micro-group*  $g$  to be a subset of sub-trajectories associated with a specific representative trajectory in  $I_i$  that satisfies the following condition:  $\forall ST_j^i \in g, ST_j^i$  is either the representative trajectory of  $g$  or a sub-trajectory which voted for it [14].

Note that a micro-group may be seen as the most primitive form of cluster of sub-trajectories. However, in this context we differentiate the two entities by means of the notion of *density-based* clustering [15] (we refer the reader to the pipeline's phase IV). NET-CUTiS needs also to monitor the evolution of

micro-groups discovered during previous time windows. To this end our approach considers typical evolution patterns, i.e., *appear, disappear, split, survive, and merge* [16]. We report that the *merge* pattern is treated as a special case, as the merging of two or more micro-groups is considered as a cluster of sub-trajectories (from this, it follows that a cluster may have multiple representative trajectories). The algorithm that NET-CUTiS employs to detect and incrementally maintain micro-groups is the same we previously introduced in [14], thus we refer the reader to that paper for more details on it.

5) *Pipeline, phase IV – sub-trajectory clustering*: the goal of the pipeline's final phase is to determine the clusters of trajectories from the micro-groups returned by the third phase. In the following we introduce some preliminary notions that underlie that of *cluster of sub-trajectories*.

**Definition 3.5: (Directly Density Reachable)** Let  $G_i$  be the set of micro-groups associated with time window  $i$ . A micro-group  $g_j \in G_i$  is said to be *directly density reachable* from a micro-group  $g_k \in G_i$  w.r.t.  $\epsilon, \delta$ , and  $\tau$ , if  $\exists ST_j^i \in g_j, \exists ST_k^i \in g_k$ , such that the following three conditions hold:

- (1)  $distance(ST_j^i, ST_k^i, \delta) \leq \epsilon.$
- (2)  $|N_\epsilon(ST_j^i)| \geq \tau$ , where  $N_\epsilon(ST_j^i) = \{ST_p^i \in \{g_j, g_k\} \text{ s.t. } distance(ST_j^i, ST_p^i, \delta) \leq \epsilon\}.$
- (3)  $|N_\epsilon(ST_k^i)| \geq \tau$ , where  $N_\epsilon(ST_k^i) = \{ST_q^i \in \{g_j, g_k\} \text{ s.t. } distance(ST_k^i, ST_q^i, \delta) \leq \epsilon\}.$

**Definition 3.6: (Density Reachable)** A micro-group  $g_i$  is said to be *density reachable* from a micro-group  $g_k$  w.r.t.  $\epsilon, \delta$ , and  $\tau$ , if there is a chain of micro-groups  $\{g_{s_1}, \dots, g_{s_n}\}$ , with  $g_{s_1} = g_i$  and  $g_{s_n} = g_k$ , such that  $g_{s_{j+1}}$  is *directly density reachable* from  $g_{s_j}$ .

**Definition 3.7: (Density Connected)** A micro-group  $g_i$  is said to be *density connected* to  $g_k$  w.r.t.  $\epsilon, \delta$ , and  $\tau$ , if there is a micro-group  $g_j$  such that  $g_i$  and  $g_k$  are *density reachable* from  $g_j$  w.r.t.  $\epsilon$  and  $\tau$ .

We report that Definitions 3.6 and 3.7 are different (indeed, the former is not symmetric). At this point we can introduce the notion of *cluster of sub-trajectories*.

**Definition 3.8 (Cluster of sub-trajectories):** Let  $G_i = \{g_1, \dots, g_n\}$  be a set of micro-groups at time window  $i$ . A *cluster*  $C$  w.r.t.  $\epsilon, \delta$  and  $\tau$  is a nonempty subset of  $G_i$  that satisfies the following two conditions:

- (1)  $\forall g_j, g_k$ , if  $g_j \in C$  and  $g_k$  is *density reachable* from  $g_j$  w.r.t.  $\epsilon, \delta$ , and  $\tau$ , then  $g_k \in C$ .
- (2)  $\forall g_j, g_k \in C, g_j$  is *density connected* to  $g_k$  w.r.t.  $\epsilon, \delta$ , and  $\tau$ .

Intuitively, a cluster represents a set of *density connected micro-groups*. From the definitions observe also that sub-trajectories not belonging to any micro-group are considered *outliers*.

Existing clustering algorithms typically verify the density connectivity of each moving object to cluster trajectories [15]. This, in turn, makes such approaches computationally expensive. NET-CUTiS, on the other hand, simplifies the clustering process and reduces computational costs by leveraging the notion of micro-group. Algorithm 1 reports the related high-level pseudo-code. NET-CUTiS first picks a micro-group and

---

**Algorithm 1:** Sub-trajectory Clustering

---

**Input:**

- The set of micro-groups at time window  $i$ ,  $G_i$ .
- Distance threshold  $\epsilon$ .
- Size threshold  $\tau$ .
- Time tolerance  $\delta$

**Output:** The set of clusters,  $C$ .

```
1 begin
2   while  $G_i \neq \emptyset$  do
3     randomly pick a micro-group  $g_i \in G_i$ ;
4      $c \leftarrow g_i$ ,  $C \leftarrow C \cup c$ ;
5      $G_i \leftarrow \{G_i \setminus g_i\}$ ;
6     foreach unvisited  $g_i \in c$  do
7       mark  $g_i$  as visited;
8       foreach  $g_k \in \{G_i \setminus g_i\}$  do
9         foreach  $ST_j^i \in g_i, ST_k^i \in g_k$  do
10          if  $\text{distance}(ST_j^i, ST_k^i, \delta) \leq \epsilon$  then
11             $n_i \leftarrow |N_\epsilon(ST_j^i)|$ ;
12             $n_k \leftarrow |N_\epsilon(ST_k^i)|$ ;
13            if  $(n_i \geq \tau) \wedge (n_k \geq \tau)$  then
14               $c \leftarrow c \cup \{g_k\}$ ;
15               $G_i \leftarrow \{G_i \setminus g_k\}$ ;
16              break;
17            end
18          end
19        end
20      end
21    end
22  end
23  return  $C$ ;
24 end
```

---

initializes it as a new cluster (lines 4 - 5). For each micro-group  $g_i$  in a cluster NET-CUTiS verifies whether there exists any other micro-group  $g_k$  that can be merged with  $g_i$  (line 8). For each micro-group  $g_k$ , NET-CUTiS verifies whether  $g_i$  and  $g_k$  are directly density reachable (lines 9 - 13): if this is true then such micro-groups are associated to the same cluster (lines 14 - 15). NET-CUTiS finally terminates by returning the set of clusters found so far (line 23).

#### IV. RELATED WORK

There exist several works that address the problem of clustering trajectories within trajectory data streams, or closely related problems. In [17] the authors propose CTraStream. Roughly speaking, CTraStream processes trajectory data streams and in order to do so it discretizes the time in intervals. Within each time interval the approach first looks for line segments in the road network that are traversed by consistent amounts of moving objects – this represents the set of so called *dense line segments*. Finally, this set is used to compute clusters of trajectories that traverse such segments. We report that, differently from NET-CUTiS, CTraStream does not monitor the evolution of movements of groups of moving objects over time, which represents a major difference with respect to our work.

CUTiS\* [14] addresses the problem of discovering and maintaining clusters of sub-trajectories from trajectory data streams with objects moving in an *Euclidean* space. The authors propose an incremental strategy that avoids recomputing clusters at regular intervals.

In [1] the authors propose FlowScan, a density-based approach that aims to find *hot routes*, i.e., *heavily trafficked* paths within a road network. To achieve this goal FlowScan considers historical trajectory data and employs a trajectory clustering algorithm that relies on the notion of *traffic density* to detect *dense* road segments, i.e., road segments traversed by a number of trajectories above a given threshold. We finally report that [18] provides a novel approach to discover hot routes in real-time.

In [19] the authors propose the TCMM framework, a set of efficient approaches which goal is to create and maintain clusters of trajectories from streams of trajectory data. We report that clustering is performed over an Euclidean space and the approach does not exploit temporal information to separate temporally unrelated trajectory, thus potentially assigning to the same cluster trajectories that span different temporal intervals.

In [2], [9] the authors propose NEAT, an approach that clusters trajectories over road networks from *historical trajectory data*. The clustering approach proposed by the authors considers (1) physical constraints that characterize road networks, (2) the spatial proximity between trajectories, and (3) traffic flows spanning consecutive road segments. Clusters returned by NEAT are made up of groups of sub-trajectories that represent dense and highly continuous traffic flows.

NETSCAN [8] considers trajectories constrained by road networks and aims at clustering trajectories from historical trajectory data. The approach operates as follows: first it determines the set of *dense paths* from historical trajectory data, where a dense path represents a path where the number of moving objects traversing it is above a given threshold  $\alpha$  – for each such path we report that NETSCAN requires the maximal density difference among its edges to not exceed a given threshold  $\epsilon$ . We report that NETSCAN does not ensure that a dense path is crossed by a specific set of moving objects, and this represents a major difference w.r.t. NET-CUTiS. Subsequently, NETSCAN clusters trajectories according to their similarity with respect to dense paths, where each trajectory is associated with a dense path only if the distance between them is lower than a given threshold  $\sigma$ . Finally, We report that NETSCAN's clustering does not take temporal information into account.

#### V. EXPERIMENTAL EVALUATION

This section presents the experimental evaluation conducted to assess the validity of NET-CUTiS in terms of clustering quality and run-time performance. We start by providing the experimental setup, followed by the experimental evaluation.

**Dataset.** For the purposes of the experimental evaluation we consider the T-DRIVE dataset<sup>1</sup> [20]. Such dataset contains one week of real-world GPS data. The subset of the dataset used throughout the experiments is limited to 4,000 trajectories and exhibits an average sampling rate of one position per minute.

<sup>1</sup><http://research.microsoft.com/apps/pubs/default.aspx?id=152883>

Symbol	Set of values	Default	Description
$\epsilon$	[0.3, 0.7], step 0.1	0.5	Distance threshold
$\tau$	[5, 25], step 5	10	Density threshold
$\delta$	[3, 7] minutes, step 1	5 minutes	Time tolerance
$\gamma$	[0.5, 0.9], step 0.1	0.7	Traj. compression threshold
Time window	[5, 7, 9] minutes	7 minutes	Time window size

TABLE I  
PARAMETERS USED BY NET-CUTiS, TOGETHER WITH THE ASSOCIATED RANGES OF VALUES USED AT RUN-TIME.

**Competitors.** To the best of our knowledge there is no incremental on-line trajectory clustering approach that addresses the problem considered in this work. The most closely related one is NETSCAN [8]. However, NETSCAN exhibits three major differences with respect to NET-CUTiS: (1) roughly speaking, each dense path represents the centroid of a cluster that groups together sub-trajectories close in space but *not necessarily* close in time; (2) NETSCAN may associate moving objects with multiple clusters; (3) NETSCAN does monitor the evolution of clusters over time. To perform meaningful comparisons we implement a modified version of NETSCAN that leverages the time discretization strategy used by NET-CUTiS.

**Run-time parameters.** Table I reports the list of parameters used by NET-CUTiS during the evaluation, together with the associated ranges of values. For what concerns the parameters used by NETSCAN, we remind that  $\sigma$  depends on  $\epsilon$ , while  $\varepsilon$  and  $\alpha$  depend on  $\tau$ .

**Experimental Methodology.** Clusters of sub-trajectories are tracked across sequences of *five* consecutive time windows, as this enables to observe how clusters evolve over a reasonably large time interval. We report that the results did not change significantly when considering larger numbers of windows. Due to space limitations we omit such experiments from the presentation.

**Evaluation Metrics.** In the absence of a ground-truth we evaluate the competitors' clustering quality by means of a pair of well-known metrics [21], i.e., *compactness* and *separation*. The goal of compactness is to establish how *closely related* moving objects making up individual clusters are. Let  $C_i$  be the set of clusters found within a time window  $i$ . The *compactness* of a cluster  $c \in C_i$  is the average pairwise spatio-temporal distance between its moving objects. As such, the overall compactness can be defined as:

$$\frac{1}{|C_i|} \sum_{c \in C_i} \frac{2}{|c|^2 - |c|} \sum_{ST_j^i, ST_k^i \in c, j \neq k} distance(ST_j^i, ST_k^i, \delta),$$

with  $|c|$  being the number of sub-trajectories making up a cluster  $c$ . Intuitively, the lower the compactness, the better the associated output.

*Separation* aims to establish *how well* clusters are *differentiated* between each other. Let  $C_i$  be the set of clusters found within a time window  $i$ . Then, separation is calculated as the

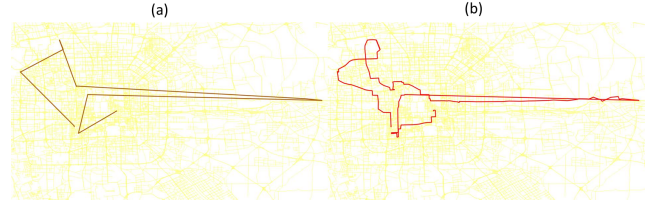


Fig. 1. Example of trajectory *densification*: plot (a) depicts the original trajectory, plot (b) its *densified* version.

average distance between items belonging to different clusters:

$$\frac{1}{\binom{|C_i|}{2}} \sum_{\substack{c_l, c_p \in C_i, \\ l \neq p}} \frac{1}{|c_l| \cdot |c_p|} \sum_{\substack{ST_j^i \in c_l, \\ ST_k^i \in c_p}} distance(ST_j^i, ST_k^i, \delta).$$

Intuitively, the higher the separation, the better the associated output. To further complement the evaluation of the clustering quality we *visually inspect* some of the results produced by NET-CUTiS – we report that visual was adopted in several past works as well [1], [2], [15], [22]. In general, visual inspection turns out to be a useful tool whenever (1) different approaches produce clusters that have different semantics, (2) different sets of parameters yield clusters which perform well in terms of quality metrics but clearly show different characteristics, or (3) when a ground truth is not available.

**Data preparation.** Data preparation consisted of two phases: *trajectory densification* and *network paths computation*. The T-DRIVE dataset presents an average sampling rate of 1 minute, which possibly represents an important issue. To overcome this problem we perform an operation called *trajectory densification* over the original trajectories, i.e., we assume that moving objects always traverse the shortest path between pairs of consecutive samples and reconstruct their trajectories accordingly. Figure 1 provides an example of the application of such technique. Finally, observe that the above choice is in line with existing literature [23], [24].

From Section III-B1 we remind that NET-CUTiS requires to find out a set of network paths from selected historical trajectory data. We know that trajectories constrained by a road network exhibit strong spatio-temporal regularity. As such, we pick up from the T-DRIVE dataset data that covers the 4th of February 2008 and extract a set of network paths accordingly. We assume that this data contains common mobility behaviors unraveling over typical weekdays (e.g., trafficked routes traversed by people commuting between home and work). We report that the extraction process yielded 833 network paths.

**Evaluation overview.** The evaluation that follows first presents an analysis concerning the clustering quality yielded by the competitors (Section V-A). To this end we study the effects that the parameters used by the two considered approaches have on the clustering quality (Table I). As some parameters are specific to NET-CUTiS, NETSCAN is considered only when applicable. The evaluation then concludes

by analyzing the run-time performance of the competitors (Section V-B).

#### A. Clustering quality analysis

**Study on the variation of the distance threshold  $\epsilon$ .** In this study we analyze how variations in the distance threshold  $\epsilon$  affect the quality of the results – to this end, we vary  $\epsilon$  in the  $[0.3, 0.7]$  range. All the other parameters are kept fixed to their respective defaults (Table I). Figure 2, first row, reports the results. On the one hand, from the Figure we observe that NET-CUTiS and NETSCAN behave similarly since they both yield high cluster separation (which is indicative of good performance). On the other hand, the compactness yielded by NETSCAN is lower (better) than that of NET-CUTiS. We believe this is due to the different amount of representative trajectories (w.r.t., dense paths) that the two approaches associate with each cluster. More precisely, NET-CUTiS possibly use multiple network paths per cluster, while NETSCAN uses only one dense path per cluster. Considering that every sub-trajectory within a cluster shares the same dense path, the average pairwise distance over the clusters returned by NETSCAN is lower than the one found over the clusters returned by NET-CUTiS. Overall, we conclude that variations in  $\epsilon$  do have minor influences on compactness and separation.

**Study on the variation of the density threshold  $\tau$ .** In this study we analyze how variations in the density threshold  $\tau$  affect the quality of the results. To this end, we vary  $\tau$  in the  $[5, 25]$  range, while keeping all the other parameters fixed to their respective defaults (Table I). Figure 2, second row, reports the results. From the Figure we observe that NET-CUTiS and NETSCAN both present high separation. For what concerns compactness, we observe that NETSCAN achieves lower measures than NET-CUTiS. However, we observe again that this is due to the different strategies used by the two approaches to build clusters. Finally, we report that when  $\tau$  increases the number of clusters returned by NET-CUTiS decreases, thus increasing the resulting compactness – note that the opposite happens when  $\tau$  decreases.

**Study on the variation of the time tolerance  $\delta$ .** In this study we analyze how variations in the time tolerance  $\delta$  affect the quality of the results returned by NET-CUTiS – to this end we vary  $\delta$  in the  $[3, 7]$  minutes range. All the other parameters are kept fixed to their defaults (Table I). As this parameter is used only by NET-CUTiS, in this batch of experiments we do not consider NETSCAN. Figure 2, third row, reports the results. From the Figure we observe that NET-CUTiS generally achieves high separation and that this measure tends to increase as  $\delta$  increases. We also report that when  $\delta$  increases the amount of sub-trajectories associated with some cluster tends to increase, which in turn increases the sum of pairwise distances between objects within a cluster. Overall, this has the effect of increasing the compactness.

**Study on the variation of the compression threshold  $\gamma$ .** In this study we analyze how variations in  $\gamma$  affect the quality of the results returned by NET-CUTiS – to this end, we vary  $\gamma$  in the  $[0.5, 0.9]$  range. All the other parameters are kept

fixed to their respective defaults (Table I). As this parameter is used only by NET-CUTiS, in this batch of experiments we do not consider NETSCAN. Figure 2, fourth row, reports the results. From the Figure we observe that NET-CUTiS achieves high separation. We also observe that when  $\gamma$  increases the number of clusters tends to increase (on top of the bars we report the average number of clusters found). Indeed, the larger  $\gamma$ , the lesser the number of neighbors retrieved when computing a range query from a sub-trajectory. Consequently, the compactness of the clusters tends to slightly decrease.

**Study on the variation of the size of time windows.** In this study we analyze how variations in the size of time windows affect the quality of the results returned by NET-CUTiS and NETSCAN. To this end we vary the size of the time windows in the  $[5, 9]$  minutes range, while the other parameters are kept fixed to their respective defaults (Table I). Figure 2, fifth row, reports the results. From the Figure we observe that NET-CUTiS and NETSCAN behave similarly in terms of separation – indeed, both approaches yield very high separation in both scenarios. We also report, however, that NETSCAN yields lower compactness than NET-CUTiS. Overall, variations in the size of time windows have minor influences on compactness. Finally, we report that using larger time windows allows NET-CUTiS to achieve lower separation, as the number of clusters that our approach can discover increases as well.

**Visual Inspection.** In this study we analyze the clusters returned by NET-CUTiS by means of visual inspection. For the purposes of this study we keep all the parameters fixed to their respective defaults (Table I). Figure 3 shows the representative trajectories of four clusters found by NET-CUTiS at the end of the first and second time windows. If we take a closer look at the locations where the four clusters fall, we observe that they tend to be located across the ramifications of two national expressways (G1 and G6) and two local expressways (S11 and S12). The most notable one is the Airport Expressway, officially the S12, that links central Beijing to the Beijing Capital International Airport. We finally observe that NET-CUTiS produces clusters having long representative routes: this may prove useful to location-based services that deal with the optimization of bus lines, or those offering ride-sharing services [25].

#### B. Run-time performance analysis

**Study on the variation of the distance threshold  $\epsilon$ .** In this study we analyze how variations in the distance threshold  $\epsilon$  affect the performance of NET-CUTiS, and compare our approach with NETSCAN. To this end, we vary  $\epsilon$  in the  $[0.3, 0.7]$  range, while keeping the other parameters fixed to their respective defaults (Table I). Figure 4, first row, reports the overall execution time of the approaches. From the Figure we observe that NET-CUTiS outperforms NETSCAN, except when considering the first time window – this is due to the initial computation of network paths. We finally observe that increasing  $\epsilon$  does not have relevant impacts on NET-CUTiS' performance, as the difference between the lowest and the highest  $\epsilon$  does not exceed 1 second.

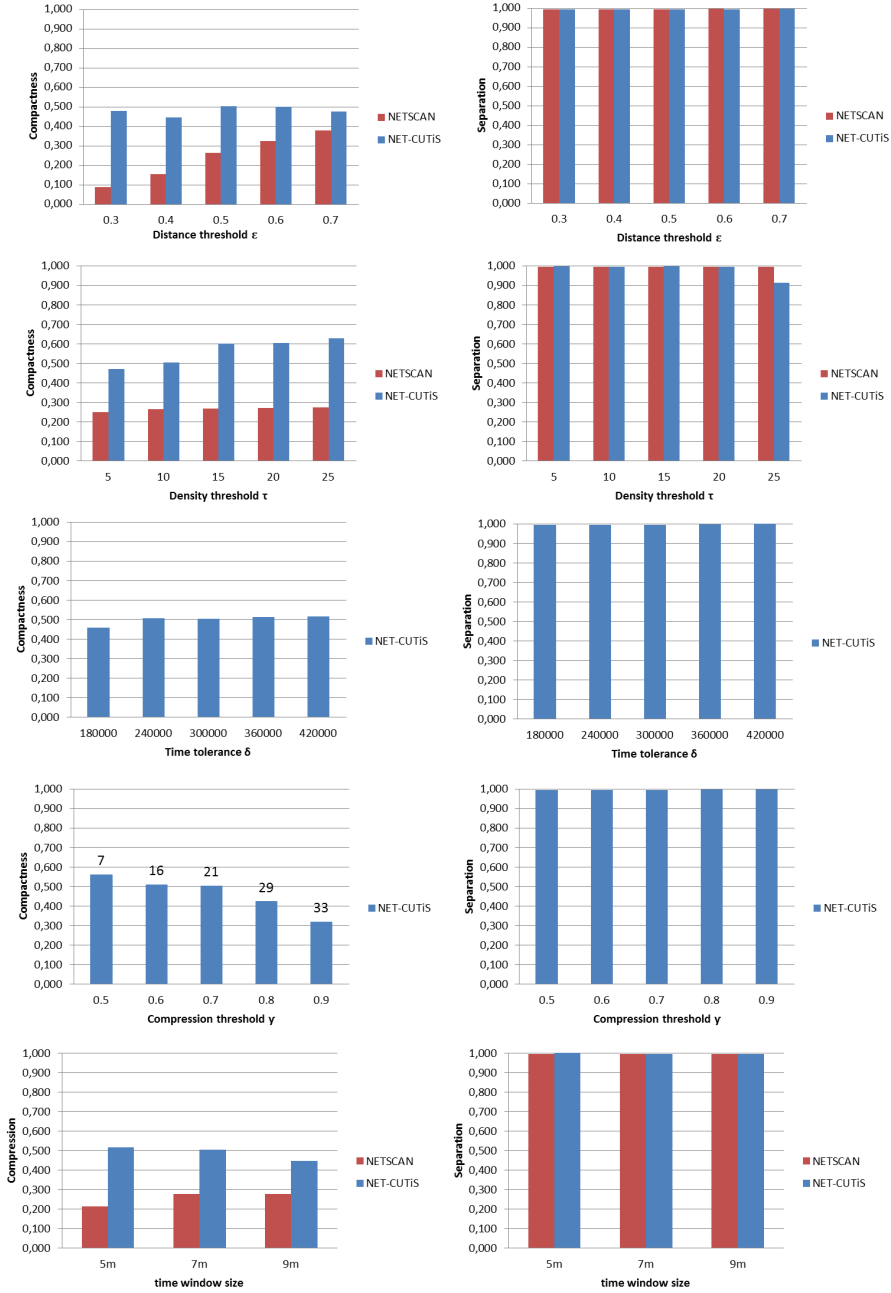


Fig. 2. Analysis of the clustering quality. *Left column*: compactness. *Right column*: separation. *First row*: distance threshold  $\epsilon$ . *Second row*: density threshold  $\tau$ . *Third row*: time tolerance  $\delta$ . *Fourth row*: compression threshold  $\gamma$ . *Fifth row*: size of time windows.

**Study on the variation of the density threshold  $\tau$ .** In this study we analyze how variations in the density threshold  $\tau$  affect the performance of NET-CUTiS and NETSCAN. To this end, we vary  $\tau$  in the  $[5, 25]$  range, while keeping the other parameters fixed to their respective defaults (Table I). Figure 4, second row, reports the overall execution time of the approaches. From the Figure we observe that NET-CUTiS outperforms NETSCAN except, again, when considering the

first time window. We also report that  $\tau$  does not exhibit a particular influence on the performance of NET-CUTiS.

**Study on the variation of the time tolerance  $\delta$ .** In this study we analyze how variations in  $\delta$  affect the performance of NET-CUTiS – in the batch of experiments that follow we do not consider NETSCAN, as this parameter is specific to NET-CUTiS. To this end, we vary  $\delta$  in the  $[3, 7]$  minutes range, while keeping the other parameters fixed to their respective



Fig. 3. Four Beijing Expressways (G1 - blue color, G6 - red color, S11 - pink color and S12 - green color) crossed by NET-CUTiS clusters. *Left column:* first time window. *Second column:* second time window.

defaults (Table I). Figure 4, third row, reports the overall execution time of the approaches. From the Figure we observe that increasing  $\delta$  does not degrade performance. Indeed, even if the number of sub-trajectories retrieved by range queries is expected to increase when the time tolerance  $\delta$  increases, the performance difference turns out to be negligible.

**Study on the variation of the compression threshold  $\gamma$ .** In this study we analyze how variations in  $\gamma$  affect the performance of NET-CUTiS – in the batch of experiments that follow we do not consider NETSCAN as this parameter is specific to NET-CUTiS. To this end, we vary  $\gamma$  in the  $[0.5, 0.9]$  range, while keeping the other parameters fixed to their respective defaults (Table I). Figure 4, fourth row, reports the overall execution time of the approaches. From the Figure we observe that the performance improves as we increase  $\gamma$  – indeed, the lowest execution time occurs when  $\gamma$  is equal to the maximum value considered (0.9). This is expected since increasing  $\gamma$  has the effect of reducing the compression of sub-trajectories.

**Study on the variation of the size of time windows.** In this study we analyze how variations in the size of the time windows affect the performance of NET-CUTiS. We vary the parameter in the  $[5, 9]$  minutes range, while keeping the other parameters fixed to their respective defaults (Table I). Figure 4, fifth row, reports the overall execution time of the approaches. From the Figure we observe that the performance of NET-CUTiS tends to improve when the size of time windows decreases, due to the resulting decreased number of sub-trajectories. Finally, we report that NET-CUTiS outperforms NETSCAN (except for the first time window).

**Final considerations.** From the experiments we observe that the clusters produced by NET-CUTiS are characterized by high (good) separation in the vast majority of cases. As for the compactness, the most relevant parameters turns out to be  $\gamma$  and  $\delta$ ; more precisely, low  $\gamma$  values increase the amount of network paths per cluster, while high  $\delta$  values tend to increase the amount of sub-trajectories associated with each cluster. Overall, in both cases we observe an increase in the average pairwise distance within clusters, which in turn ends up affecting negatively the compactness. For what concerns run-time performance, the most relevant parameters are  $\gamma$

and the size of the time windows. Also, when the size of the time windows increases, the number of sub-trajectories generally increases. In both cases there is a negative effect on the performance, since the amount of trajectories to process increases. We thus suggest to pick large  $\gamma$ , small  $\delta$  and small size of time windows, as we argue this represents a reasonable trade-off between performance and quality.

To discover density based clusters from trajectory data streams over road networks NET-CUTiS employs the same algorithm previously introduced by the authors in [14]. Such algorithm consists of two steps, namely, the one dealing with the creation and maintenance of micro-groups, and the one dealing with clustering. We remind that the clustering step dominates the algorithm's total cost due to the necessity of merging micro-groups into clusters (which represents a very expensive operation). The performance evaluation of these steps is already available in [14], thus we omit it here due to lack of space.

## VI. CONCLUSIONS

In this paper we considered the problem of detecting and maintaining clusters of data stream constrained by road networks. The problem is particularly relevant, as today's vast availability of trajectory data that can be collected, monitored, and analyzed gives the opportunity to tackle relevant mobility challenges in urban settings. To address the problem we propose NET-CUTiS, a novel on-line incremental clustering approach. In the extensive experimental evaluation we assess the validity of NET-CUTiS and compare it with a modified version of NETSCAN, both in terms of quality of results and run-time performance. From the results we observe that NET-CUTiS is comparable to NETSCAN in terms of quality while, at the same time, outperforming it in terms of performance.

As a possible line of future research we aim at investigating novel and more efficient strategies to discover network paths, as well as to determine when network paths have to be recomputed when they do not reflect any more ongoing traffic behaviors. Furthermore, clustering algorithms generally require to provide a set of parameters, and trajectory clustering algorithms such as NET-CUTiS make no exception [2], [7]–[9], [15]. Thus, another potential line of future research would

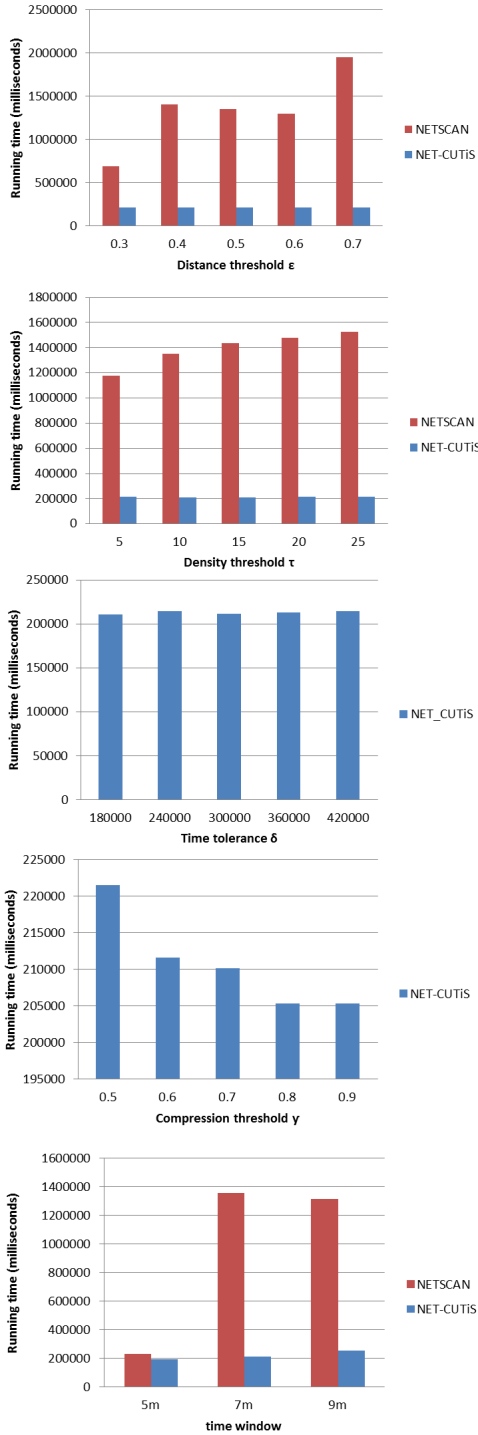


Fig. 4. Run-time performance analysis. *First row:* distance threshold  $\epsilon$ . *Second row:* density threshold  $\tau$ . *Third row:* time tolerance  $\delta$ . *Fourth row:* compression threshold  $\gamma$ . *Fifth row:* size of time windows.

attempt to automatize the optimization of such parameters and we conjecture this is possible by leveraging machine learning

tools for hyper-parameter training.

#### ACKNOWLEDGMENTS

Research partially supported by NSERC Canada, FUNCAP SPU 8789771/2017 and UFC-FASTEF 31/2019.

#### REFERENCES

- [1] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. Traffic density-based discovery of hot routes in road networks. In *SSTD*. 2007.
- [2] Binh Han, Ling Liu, and Edward Omiecinski. Neat: Road network aware trajectory clustering. In *ICDCS*, pages 142–151, 2012.
- [3] Lu-An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Chih-Chieh Hung, and Wen-Chih Peng. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012.
- [4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998.
- [5] Christian S. Jensen, D. Lin, and Beng-Chin Ooi. Continuous clustering of moving objects. *IEEE TKDE*, 19, 2007.
- [6] Xiaohui Li, Vaida Ceikute, Christian S Jensen, and Kian-Lee Tan. Effective online group discovery in trajectory databases. *TKDE*, 25:2752–2766, 2013.
- [7] Lu Chen, Yunjun Gao, Ziquan Fang, Xiaoye Miao, Christian S. Jensen, and Chenjuan Guo. Real-time distributed co-movement pattern detection on streaming trajectories. *Proc. VLDB Endow.*, 12(10):1208–1220, 2019.
- [8] Ahmed Kharrat, Iulian Sandu Popa, Karine Zeitouni, and Sami Faiz. Clustering algorithm for network constraint trajectories. In *Headway in Spatial Data Handling*, pages 631–647. Springer, 2008.
- [9] Binh Han, Ling Liu, and Edward Omiecinski. Road-network aware trajectory clustering: Integrating locality, flow, and density. *IEEE Transactions on Mobile Computing*, 14(2):416–429, 2015.
- [10] Yu Zheng. Trajectory data mining: an overview. *TIST*, page 29, 2015.
- [11] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, and Ahmed Kharrat. Spatio-temporal compression of trajectories in road networks. *Geoinformatica*, 19(1):117–145, 2015.
- [12] Mirco Nanni. Clustering methods for spatio-temporal data. *PhD thesis, CS Dept., University of Pisa*, 2002.
- [13] Costas Panagiotakis, Nikos Pelekis, Ioannis Kopanakis, Emmanuel Rasmasso, and Yannis Theodoridis. Segmentation and sampling of moving object trajectories based on representativeness. *TKDE*, pages 1328–1343, 2012.
- [14] T. Coelho da Silva, K. Zeitouni, and J. de Macêdo. Online clustering of trajectory data streams. In *MDM*, 2016.
- [15] Martin Ester, Hans-Peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.
- [16] Myra Spiliopoulou, Irene Ntoutsis, Yannis Theodoridis, and Rene Schult. Monic: modeling and monitoring cluster transitions. In *12th ACM SIGKDD*, pages 706–711. ACM, 2006.
- [17] Y. Yu, Q. Wang, X. Wang, H. Wang, and J. He. Online clustering for trajectory data stream of moving objects. *ComSIS*, 2013.
- [18] G AM Gomes, E Santos, C A Vidal, T L Coelho da Silva da Silva, and Jose A F Macedo. Real-time discovery of hot routes on trajectory data streams using interactive visualization based on gpu. *Computers & Graphics*, 2018.
- [19] Zhenhui Li, Jae-Gil Lee, Xiaolei Li, and Jiawei Han. Incremental clustering for trajectories. In *DASFAA*, pages 32–46, 2010.
- [20] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *18th SIGSPATIAL*, pages 99–108. ACM, 2010.
- [21] Alessandro Lulli. Distributed graph processing: Algorithms and applications. 2017.
- [22] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, pages 593–604, 2007.
- [23] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1144–1155. IEEE, 2012.
- [24] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2012.
- [25] Ece Kamar and Eric Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. In *IJCAI*, volume 9, page 187, 2009.