**Noname manuscript No.**
(will be inserted by the editor)

# AntClu: Any-time Trajectory Clustering via Dynamic Trend Representation

**Yi Zhao · Chongming Gao · Ruizhi Wu · Qinli Yang · Junming Shao**

Received: date / Accepted: date

**Abstract** In recent years, clustering trajectory data has been extensively explored to discover similar patterns of moving objects. Existing approaches, often cluster whole life-span trajectories into several groups according to some trajectory similarities such as dynamic time warping and edit distance. However, the trajectory of a given moving object is dynamic and evolved over time. Exploring the dynamic grouping patterns of moving objects (e.g., the expanding, shrinking, emerging or disappearing of clusters) over time thus offers a more dedicated venue to analyze the evolved moving patterns. To address this problem, in this paper, we propose a new any-time trajectory clustering algorithm, called AntClu, building upon the concepts of automatic dynamic trend representation and density-based online clustering. The basic idea is to learn a dynamic representation for each trajectory to capture "current trend", and then cluster these "trends" in an online setting. Therefore, AntClu is capable of clustering trajectories at any time, and time-changing clusters are available whenever the request comes. More importantly, unlike traditional data stream clustering approaches or online learning, AntClu is also independent of time-window. The experimental results on real-world data sets further demonstrate its effectiveness and efficiency.

**Keywords** Trajectory clustering · Anytime clustering · Dynamic representation

## 1 Introduction

During the past decade, we have witnessed how the popularity of mobile equipment, especially smart phones embedded with advanced GPS sensors, have facilitated the acquisition of real-time trajectory data (Zheng et al, 2014; Zheng, 2015). Mining these trajectory data has gained increasing attention in

University of Science and Technology of China, Sichuan, China

diverse fileds such as intelligent traffic management, instant POI recommendation, moving pattern discovery, to mention a few (Won et al, 2009; Han et al, 2012; Wu et al, 2018b). As the primary task of spatio-temporal data mining, clustering provides a good way to summarize the trajectory data, and allows discovering objects sharing similar moving patterns (Phang et al, 2003; Lee et al, 2007; Hu et al, 2013a). Moreover, trajectory clustering is also beneficial to many other related mining tasks such as event detection (Piciarelli and Foresti, 2006), pattern mining (Mohammed et al, 2009) and location prediction (Monreale et al, 2009; Wu et al, 2018a).

To present, many trajectory clustering algorithms have been proposed (Nanni and Pedreschi, 2006; Li et al, 2010b; Lee et al, 2007; Yanwei et al, 2013; Mao et al, 2016). A typical approach is to treat the whole trajectory as an object, and then cluster them into several groups with different similarity measures. However, due to the unique property of moving objects, the associated trajectories of moving objects are dynamic and change over time. Therefore, clustering the whole life-span trajectory fails to capture the time-changing grouping patterns. For instance, the moving trajectory of a person could be clustered with his co-workers in the daytime while with another group with his family members at night. It is necessary and important to discover these dynamic clusters over time. Nevertheless, quite a few relevant work has exploited the problem.

The mainstream of recent studies treat a given trajectory data as a data stream, and then employ time-window data stream algorithm to perform a two-phase clustering: online clustering and offline clustering. Sliding time-window is one of the broadest-used models for clustering trajectory data (Mao et al, 2016; Yanwei et al, 2013; da Silva et al, 2016). By specifying a given fixed-size window, the sub-trajectories in the window are used to represent the current trajectories and are formed as many online clusters (i.e., micro-clusters). When a clustering request comes, an offline clustering such as k-means or DBSCAN is to perform to yeild more than a skeletal result. Such simple modification in the context of data stream mining cannot fit the trajectory data clustering since the sub-trajectory in a sliding window is insufficient to represent "current trend" of a trajectory. Selecting a suitable window size is also a nontrivial task in real-world scenarios since different moving objects often have different shifting time. Besides, unlike newly arriving objects in traditional data streams, trajectory clustering always works on the same dynamic trajectories except that new moving objects are involved.

In this paper, we introduce a new any-time trajectory clustering approach, called AntClu, which is built upon the dynamic trend representation and density-based data stream clustering. As illustrated in Fig. 1, we aim to discover trajectory clusters at any time in an evolving trajectory data stream, instead of using a window to capture the "current trend" (Fig. 1(b)). Thus, an any-time trend representation is proposed (Fig. 1(c)). Based on the derived representation, the trajectory clusters are easily obtained online. Apparently, our work can better capture the evolution of the trajectories. The contributions of the paper are summarized as follows:

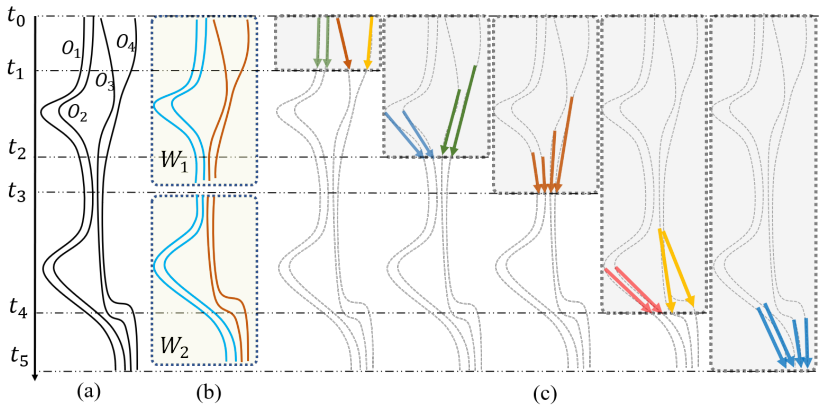



**Fig. 1** The framework of AntClu algorithm. (a) The trajectories of four objects during $t_0$ to $t_5$; (b)Trajectory representation and consequent clustering results via time-window based model; (c) Proposed dynamic trend representation and the derived trajectory clusters at five time points.

- **Dynamic Trend Representation.** Unlike time-window based trajectory representation, AntClu introduces a dynamic trend representation to capture the current trend of a trajectory at any given time point. The *trend*, which is the core of our work, can effectively maintain a synopsis of recent movement of objects. Beyond, we propose a new distance function of line segment as well as an error bounding algorithm to make the *trend* to be more smooth and representative.
- **Any-time Clustering.** Relying on dynamic trend representation, AntClu leads itself to support any-time clustering. Two different types of clusters, referred to as active-cluster and inactive-cluster, are created and maintained online, and the pruning strategy ensures that the evolving clusters can be obtained effectively.
- **High Performance.** Our experiment results on real-world data sets show that *AntClu* is insensitive to noises and can perform robustly with various parameters and data sets.

The rest of the paper is organized as follows: Section 2 surveys the related work. Section 3 introduces the fundamental concepts of our framework. In Section 4 we present how to maintain the trends of trajectories and in Section 5 the general framework and the key algorithms are given. Finally, Section 6 shows a satisfactory experimental outcome of our work, in terms of both effectiveness and efficiency.

## 2 Related Work

Trajectory clustering has gained increasing attention in recent years, and many clustering algorithms have been extended for clustering trajectory data(Han

et al, 2015; Deng et al, 2015; Hu et al, 2013b; Chen et al, 2013). Here we only survey some closest works.

**Trajectory Clustering on Static Databases** (Nanni and Pedreschi, 2006) propose an adaptation of OPTICS for trajectory data, based on a time-focused distance metric between trajectories. However, from real world experience, trajectories have different patterns during different time intervals. As a result, (Lee et al, 2007) turn to exploit the common sub-structures of the trajectories. They present a framework called TraClus, which partitions the trajectory into line segments based on minumum description length principle and then respectively groups them into sub-trajectory clusters based on density. Despite the salient accomplishment that TraClus have made, it suffers from high computation cost and cannot be applied in online clustering due to the limited memory and computing resources.

**Continuous Trajectory Clustering** (Jensen et al, 2007) proposes a new scheme that is capable of incrementally clustering moving objects and extends the similarity metric by taking semantic factors into account. In that paper, the distance between two objects is the weighted summation of euclidean distances between the trajectory points respectively and the closer they are to the current time, a higher weight is assigned. (Li et al, 2010b), following the similar idea of (Lee et al, 2007), i.e. 'segmentation-and-clustering', introduce a micro-/macro-clustering method, TCMM, to handle the trajectory data incrementally. But it still cannot process streaming data, because sub-trajectory micro-clustering has to wait for enough new points to accumulate to form sub-trajectories, which needs additional buffer space and waiting time, let alone the expensive computing cost of MDL-based trajectory simplification methods that are inherited from TraClus.

**Online Trajectory Clustering.** The aforementioned approaches mainly focus on static data set and could not be easily applied to real-time scenarios. To this end, (Silva et al, 2016) define a new structure called Micro-Groups, which can address the problem of online discovery of mobility patterns and maintain the pattern evolution by tracking the sub-trajectories of moving objects. (Yanwei et al, 2013) propose a framework, supporting to incrementally process line-segments at each time span and online update clusters based on a bi-tree index. (Mao et al, 2016) propose a sliding-window based model and collect sub-trajectory aggregation by various Minimum Bounding Rectangles (MBRs). With adoption of two novel synopsis data structures, the Temporal Trajectory Cluster Feature (TF) and the Exponential Histogram of Temporal Trajectory Cluster Feature (EF), the model enables the features of clusters to shift towards the fresh sub-trajectories, in which way it can efficiently handle the evolution both in interior and exterior of the clusters. (da Silva et al, 2016) present CUTiS, an online trajectory stream clustering approach, to maintain sub-trajectory clusters in each time window. Especially, they use a voting scheme to set the center of the cluster instead of the average features of cluster members. In conclude, those work is indeed capable of online clustering, whereas most of them are methodically limited by the inherent defect of sliding-window model — selecting a proper size of the time window. More-

over, the time window prevents it from exploiting an any-time clustering over trajectory stream, which is one of our contribution and will be discussed in following sections of this paper.

**Moving Together Pattern Mining.** The works over Moving-Together Pattern Mining and Travelling Companions Discovery (Tang et al, 2012; Gudmundsson and van Kreveld, 2006; Jeung et al, 2008; Li et al, 2010a) is similar to trajectory clustering to some extent. Such kind of work is intended to discover moving objects that are excepted to stay together for a while during the motion, and can be concluded as the search for flocks((Gudmundsson and van Kreveld, 2006), convoy(Jeung et al, 2008) and swarms(Li et al, 2010a), etc. (Li et al, 2004), for example, propose the concept of moving micro-cluster (MMC for short). An MMC maintains a profile at each time point that records the average coordinates and velocity of cluster members, aiming to capture the moving patterns those objects share. By defining the bounding boxes of clusters, the algorithm can detect the split and collision events of clusters. Although those work is not that related to our work, they are heuristic in trajectory clustering and are worthwhile to be mentioned.

## 3 Problem Statement

Let $\mathcal{S}$ denotes a trajectory data stream. Generally, it's assumed unbounded, containing infinite location points of moving objects. The snapshot of the stream at time $t$ is a batch of location points represented as $\mathcal{S}(t) = \{p_1^{(t)}, p_2^{(t)}, \ldots\}$, where $N$ is the total number of moving objects at time $t$. Each point $p$ contains the 2-dimensional GPS coordinate, attached with a timestamp $t$ and object $id$. Formally, the trajectory $\mathcal{T}$ of a moving object is defined as follows.

**Definition 1 (Trajectory)** The trajectory of the $i$-th moving object $\mathcal{T}_i$ is denoted as $\{p_i^{(t_0)}, p_i^{(t_1)}, \ldots\}$, i.e., a sequence of the location points stamped by their arrivaling time.

For better expression of the movement, we denote $\mathcal{T}$ as a sequence of *trajectory line segments*, defined as follows.

**Definition 2 (Trajectory Line Segment)** A trajectory line segment $L^{(t_i,t_j)}$ is the direct line connecting two points $p^{(t_i)}$ and $p^{(t_j)}$. $L^{(t_j)}$ is used to represent $L^{(t_i,t_j)}$, if $t_i$ and $t_j$ are two adjacent timestamps.

The any-time trajectory clustering in this paper is to dynamically group similar trajectories $\mathcal{T} = \{L^{(t_1)}, L^{(t_2)}, \ldots, L^{(t_n)}\}$ into the same clusters $TC_k \in \boldsymbol{TC}(t_n)$ at any given time $t_n$, where $\boldsymbol{TC}(t_n)$ the set of the clusters at that time.

To this end, three challenges should be addressed. (1) When the trajectory data points continuously arrive, a dynamic representation should be introduced to represent the current trend of a given trajectory. This is different from traditional way where the moving object is usually represented by the

freshest location point or the sub-trajectory in a time window. (2) A new similarity measure should be proposed to support dynamic trend representation since we do not have the entire trajectory in advance. (3) An any-time trajectory clustering strategy should be introduced based on the dynamic trend representation.

## 4 Dynamic Trend Representation

In the streaming scenarios, the trajectory of a given moving object keeps growing. As a result, it's barely impossible to consider the whole life-span trajectory but the sub-trajectory. In traditional way, the sub-trajectory is often framed by a time-window or is constrained by an amount of data points. Since the size of time-window is pre-defined, it can neither well represent the recent movement of the object nor adapt to the evolution in the trajectory stream (cf. Fig. 1). In this section, we introduce a novel data structure, *trend*, which is capable of dynamically maintaining a sub-trajectory to represent current moving trend of a given object.

### 4.1 Instantaneous Trajectory Trend

As the moving direction or the velocity of an object often changes smoothly or rapidly over time, it is important to derive a good representation in the context of trajectory stream. Here, we introduce *Instantaneous Trajectory Trend*, abbreviated as "trend", to dynamically maintain a comparatively stable sub-trajectory. Formally, we define the trend as follows.

**Definition 3 (Trend)** Given a current trajectory $\mathcal{T}_i = \{L_i^{(t_1)}, L_i^{(t_2)}, \ldots, L^{(t_n)}\}$ of a moving object, an *instantaneous trajectory trend* is defined as a relatively stable sub-trajectory, $TR_i = \{L^{(t_m)}, L^{(t_{m+1})}, \ldots, L^{(t_n)}\}(1 \leq m \leq n)$. The trend, is further approximated as a representative line segment, denoted as $repL = L^{(t_m, t_n)}$, by connecting the starting point and ending point, where the error introduced by the approximation is within a given tolerance $\delta$.

The concept *trend* allows capturing the current trajectory moving trend. Namely, if the recent movement of an object is stable, more consecutive sub-trajectories will be used to represent the current moving trend. If the moving pattern changes, previous moving trajectory segments should be ignored and only the current moving trajectory is used. Since the sub-trajectories contained in the trend trajectory is smooth, and thus we can use the representative line segment to reveal the current moving trend efficiently. For example, Fig. X gives an illustration,

To guarantee the precision of representative line segment, we propose a distance metric to measure the deviation between an instantaneous trajectory trend and its approximated representative line segment as follows.

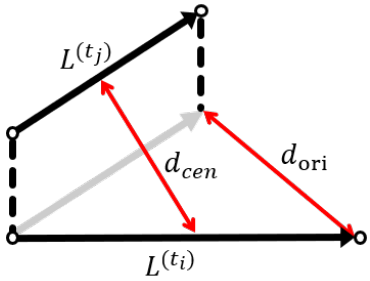



**Fig. 2** Illustration of line segment distance. The distance between two line segments is a weighted summation of $d_{cen}$ and $d_{ori}$.
Has not been referred to in the article

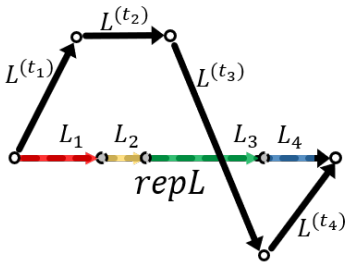



**Fig. 3** Illustration of the time synchronized line distance. Here the solid line represents the raw sub-trajectory, while the dotted line indicates the representative line segment of the trend.

## 4.2 Line Segment Distance

We start with the definition of line segment distance metric that we use not only for the error measurement of *trend* representation, but also in the clustering phase that will be detailed in Section 5. The line segment distance function has two aspects, that is the center distance and the orientation distance.

**Definition 4 (Center Distance)** The center distance is the measurement for the gap between two line segments, indicating how far the two line segments are. Let $d_{cen}$ be the center distance between two line segments $L^{(t_i)}$ and $L^{(t_j)}$, and is defined as the euclidean distance between their center points.

$$dist_{cen}(L^{(t_i)}, L^{(t_j)}) = ED(L^{(t_i)}.center, L^{(t_j)}.center) \tag{1}$$

where ED stands for the euclidean distance.

**Definition 5 (Orientation Distance)** The orientation distance $d_{ori}$ serves as an indication of how different the two line segments orient and how far they have extended in that direction. Specifically, it equals the distance between the two end points when the two start points are aligned together, and let $\overrightarrow{L}$ denote the vector that constructed by its start and end point, we can obtain the orientation distance as follows:

$$d_{ori}(L^{(t_i)}, L^{(t_j)}) = |\overrightarrow{L^{(t_i)}} - \overrightarrow{L^{(t_j)}}| \tag{2}$$

where $|\cdot|$ means the length of the vector

**Definition 6 (Line Segment Distance)** The distance between two line segements $L^{(t_i)}$ and $L^{(t_j)}$ is defined as the weighted summation of the two components:

$$dist(L^{(t_i)}, L^{(t_j)}) = \alpha * d_{cen}(L^{(t_i)}, L^{(t_j)}) + (1-\alpha) * d_{ori}(L^{(t_i)}, L^{(t_j)}) \tag{3}$$

where $\alpha = \frac{1}{1+e^{-\cos(\theta)}}$ and $\theta$ is the angle between the line segments $L^{(t_i)}$ and $L^{(t_j)}$, which can be derived as follows:

$$\cos(\theta) = \frac{\overrightarrow{L^{(t_i)}}\overrightarrow{L^{(t_j)}}}{|L^{(t_i)}||L^{(t_j)}|} \tag{4}$$

$\alpha$, served as a penalty for the line segments that oriented in dramatically different directions, is supposed to have a larger value of the line segment distance when the orientation distance is far more than the center distance, e.g. two objects moving in the same trace but heading in the opposite direction, compared with a setting of the equal coefficients. That is to say, the line segment distance is expected to assign more similarity for those with close lengths and directions. In addition, the function $f(\theta) = \frac{1}{1+e^{-\cos(\theta)}}$ decrease monotonically between 0 to 180° and can be simply proved that have a minimun at $\theta = 180°$.

### 4.3 Time Synchronized Line Distance

Building upon the line segment distance, we further introduce how to compute the Time Synchronized Line Distance (abbr. TSLD), which is used to measure the error of the approximated representative line segment.

**Definition 7 (Time Synchronized Line Distance)** For a trend $TR = \{L^{(t_m)}, L^{(t_{m+1})}, \ldots, L^{(t_n)}\}(1 \leq m \leq n)$ composed of $n - m + 1$ line segments, the error that brought in by representing it as a line segment, $repL$, is defined as below:

$$\mathcal{E}_{tsld}(TR) = \sum_{L^{(t_i)} \in TR} dist(L^{(t_i)}, L_i) \tag{5}$$

where $L_i$ is the part of $repL$, the length of which is determined by the proportion of corresponding travel time in $TR$, and $dist$ is line segment distance as proposed in Section 4.2

Fig. 3 has illustrated an intuitive example of $TSLD$, where a sub-trajectory (trend) with 4 line segments has been represented as $repL$. The lenth of each original line segment $L_i$ is only concerned with the time span between its start and end point. In other word, the longer $L^{(t_i)}$ is, the more length $L_i$ would be assigned. Afterwards, the error distance is computed as the summation of all line segment distances between each raw segment $L_i$ in trend and partition $L_i$ in $repL$. As the result, the complexity of computing tsld is $O(n)$.

When there is a rapid change of movement, the $\mathcal{E}_{tsld}$ will increase significantly, and thus the trend should be updated accordingly. Here we use $\delta$ as a parameter to control the maximum tolerance of the error. In empirical experiments, we have demonstrated that the parameter is not sensitive to our results.

Relying on the time synchronized line distance, finally we can use the representative segment to dynamically represent the moving trend of a given object.

**Algorithm 1** Trend Update Procedure

**Input:** the arriving point $p_{id}^{(t_{n+1})}$
**Output:** the matched trend $TR_{id}$

Assign $p_{id}^{(t_{n+1})}$ into its corresponding trend $TR_{id}$
**if** $TR_{id}$ is not exsit **then**
  Create a new trend $TR_{id}$ with single point $p_{id}^{(t_{n+1})}$
  **return** $TR_{id}$
**end if**
Assign $p_{id}^{(t_{n+1})}$ to $TR_{id}$
**if** $TR_{id}$ has only one line segment(i.e. two points) **then**
  **return** $TR_{id} : \{L^{(t_{n+1})}\}$
**end if**
Calculate the Time Synchronized Line Distance of $TR_{id}$
**if** $\mathcal{E}_{tsld}(TR_{id}) \leq \delta$ **then**
  $TR_{id} : \{L^{(t_m)}, \ldots, L^{(t_{n-1})}, L^{(t_n)}\}$
**else**
  $TR_{id} : \{L^{(t_{n-1})}, L^{(t_n)}\}$
**end if**
**return** $TR_{id}$

Specifically, for each point arriving at the next timestamp in the stream, we assign it to the corresponding trend according to its object-id. Afterward, the $\mathcal{E}_{tsld}$ will be calculated, and the trend will be updated if the value of $\mathcal{E}_{tsld}$ exceeds the given threshold $\delta$. The update procedure is futher given in Algorithm 1.

According to Algorithm 1, each arriving data point will be matched and incorporated into the *trends*. The trend update procedure ensures that the sub-trajectory within each *trend* is smooth so that a line segment representation will not cause too much deviation. Consequently, we can use the representative line segments for online clustering, which will be introduced in the next section. Also, we randomly sample several sub-trajectories from real world data sets and evaluate the effectiveness of trend representation based on *TSLD*. The outcome, as illustrated in Fig. 5, shows the trend representation do sense the dramatic change of the trajectory and only maintain the smoothest part of it, which proves that our method is quite convincing.

## 5 Any-time Trajectory Clustering via Dynamic Trend Representation

In this section, we introduce a new trajectory clustering method, called AntClu, which is capable of grouping the similar trajectories at any given time. The basic idea is to use the dynamic trend representation to maintain moving patterns, and clustering these trends online from the density point of view. Thereby, in the following, we start with reformatting some notions of density in the context of trajectory streams.

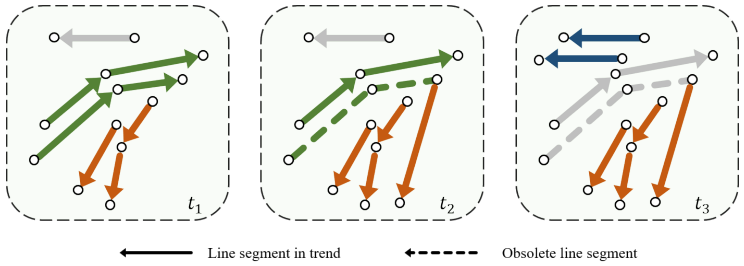



**Fig. 4** The *trends* maintain the current movement of the objects, and then are clustered according to their representation. In this figure, line segments with the same colors are in the same active trend clusters, whereas the grey stands for the inactive trend clusters. The demonstration indicates the *trend* can capture the sharp changes in movement, as well as the role exchanges between two kinds of *trend clusters*.To be referred to in article

### 5.1 Density-based Trend Clustering

We now present our density-based online clustering algorithm. In the static environment, density-based clustering can find the clusters with arbitrary shape and has been fully exploited in line segment clustering (Lee et al, 2007; Li et al, 2010b). Hence, we only clarify the necessary notions before introducing our work, in which we shared the similar concepts with other work. Formally, let $\mathcal{TR}$ denotes the set of all *trends* maintained at time $t$.

**Definition 8 ($\epsilon$-neighborhood)** The $\epsilon$-neighborhood $N_\epsilon$ of given trend $TR_i \in \mathcal{TR}$ is defined by $N_\epsilon(TR_i) = \{TR_j \in \mathcal{TR} | dist(TR_j.repL, TR_i.repL) \leq \epsilon\}$. Here $dist(x, y)$ is a metric distance function, and here the line segment distance is used.

**Definition 9 (Core Trend)** A trend $TR_i \in \mathcal{TR}$ is called a core trend w.r.t. $\epsilon$ and $\mu$ if the total weight of trends in $N_\epsilon$ is larger than a given value $\mu$.

**Definition 10 (Trend Cluster)** A trend cluster $TC$ w.r.t $\epsilon$ and $\mu$ at a any given time $t$ is a set of trends $\{TR_1, TR_2, \ldots TR_N\} \in \mathcal{TR}$. Let $t$ denotes the current timestamp, the trend cluster maintains the following features.

- $LS$: The weighted linear sum of representative line segments of incorporated trends, i.e. $LS = \sum_{TR_i \in \mathcal{TR}} f(t - t_i) TR_i.repL$, where $t_i$ is the timestamp when the $repL$ is updated upon a new arriving point.
- $w$: The overall weight of the total line segments in the cluster, i.e. $w = \sum_{i=1}^{n} f(t - t_i)$, where $n$ is the amount of line segments in the cluster.
- $N$: The amount of all *trends* in the cluster. Obviously, $N \leq n$.
- $\tau$: The activation indicator of the cluster, denoted as $\tau = \sum_{i=1}^{N} f(t - t_i)$, where $t_i$ stands for the latest updating timestamp of $TR_i$.

The decay function $f(\cdot)$ decrease the weight of each representative line segment in the trend cluster, and in this paper, we just define it as $f(\Delta t) = \lambda^{\Delta t}$, where $\Delta t$ is the time span since the line segment was incorporated and $\lambda \in (0, 1)$ is the decay rate. Such mechanism is supposed to gradually forget the effect of past data, so that the clusters are dominated by new data.

In order to calculate the distance between a trend to a trend cluster, we define the center of $TC$ in the form of a line segment and denoted as $TC.cenL = \frac{LS}{w}$. For any $TR \in TC$, $TR \in N_\epsilon(TC.cenL)$. Notice when a trend is to be incorporated into the trend cluster $TC$, it's the representative line segment that is used to update the features of $TC$.

**Lemma 1** *The features of a trend cluster could be maintained incrementally.*

*Proof* Consider a trend cluster $TC = \{LS, w, N, \tau\}$, if no trend has been merged for a time interval $\Delta t$, $TC = \{f(\Delta t) \cdot LS, f(\Delta t) \cdot w, N, f(\Delta t) \cdot \tau\}$. Otherwise, when a new *trend* $TR$ is merged by $TC$, $TC = \{LS+TR.repL, w+1, N+1, \tau+1\}$.

In other hand, the activation $\tau$ is the key feature of a trend cluster. It usually initialized as the same value of the number of trends $N$, whereas it can be gradually decayed the trends are no longer updated. If the trends keep a tight update frequency, the activation of a trend cluster is supposed to have a high value but less than $N$.

### 5.2 AntClu Algorithm

In the stream environment, a given trajectory can be viewed as a cluster object or outlier over time. Therefore, like DenStream (Cao et al, 2006), we maintain two types of clusters: active cluster and inactive cluster, respectively, depending on whether its activation $\tau \geq \mu$ or $\tau < \mu$.

**Definition 11 (Active Trend Cluster)** Active trend cluster $TC_{ac}$ is the trend cluster that has sufficient trends in $N_\epsilon(TC_{ac}.cen)$, i.e $\tau \geq \mu$.

**Definition 12 (Inactive Trend Cluster)** Inactive trend cluster $TC_{in}$ is the trend cluster that has insufficient trends in $N_\epsilon(TC_{ac}.cen)$, i.e $\tau < \mu$.

Upon the arrival of each point, it mainly involved two steps. Firstly, we update the corresponding trend $TR_{id}$ associated with the point $p_{id}^{(t_i)}$ via Algorithm 1. Afterwards, the trend $TR_{id}$ is assigned to its closest active cluster by computing the distances between the representative line segment of the trend and the centers of trend clusters. If the minimum distance is below $\epsilon$, merge $TR_{id}$ to its closest active cluster. Otherwise, we try to find its closest inactive cluster. Similarly, if the distance is below $\epsilon$, merge $TR_{id}$ to the closest inactive cluster. In addition, if the $\tau$ of the inactive cluster is above $\mu$ after merging $TR_{id}$, mark it as a new active cluster. Finally, the pseudocode of trend merging is described in Algorithm 2.

The complexity of merging fuction is $O(m * n)$, where m is the number of trend clusters. In the worst cases, a new trend has to search the entire cluster candidates before being merged. In other words, the cost of trend merging procedure is mainly concerned with the number of trend clusters at current time. Thus, it's important to limit the cluster amount.

**Algorithm 2** Trend Merge Strategy

**Input:** the trend $TR$
**Output:** the trend cluster $TC$

Find the closest active cluster $TC_{ac}$ for the trend $TR$
**if** $dist(TR.repL, TC_{ac}.cenL) \leq \epsilon$ **then**
  Merge $TR$ into $TC_{ac}$
  Update Cluster Features of $(TC_{ac})$
  **return** $TC_{ac}$
**else**
  Find the closest inactive cluster $TC_{in}$
  **if** $dist(TR.repL, TC_{in}.cenL) \leq \epsilon$ **then**
    Merge $TR$ into $TC_{in}$
    Update Cluster Features of $(TC_{in})$
    **if** $TC_{in}.\tau > \mu$ **then**
      Re-label $TC_{in}$ as an active cluster
    **end if**
    **return** $TC_{in}$
  **else**
    Create new inactive cluster $TC_{new}$ with $TR$
    **return** $TC_{new}$
  **end if**
**end if**

*Cluster Pruning.* To guarantee a safe searching space for merging new trends, and release the memory space as well, we periodocally check the activation of each trend cluster and delete them if meet the pruning conditions. We claim an active trend cluster $TC_{ac}$ is out of activity when its activation $\tau$ is below $\mu$. We specify the checking period by considering the boundary conditions where the lowest activation $\tau = \mu$. Supposing an active clsuter $TC_{ac} = \{LS, w, N, \tau\}$, of which the activation $\tau = \mu$, doesn't have a trend merged in a period $\Delta t$ and the activation $\tau$ just meets the low-bound $\beta\mu$, i.e., $\lambda^{\Delta t}\beta\mu + 1 = \beta\mu$. In that way we can obtain the minimum check period.

**Proposition 1** *The minimum time span of check the activation of active clusters is* $\mathcal{T}_{ck} = \log_\lambda \frac{\mu-1}{\mu}$*, where* $\lambda$ *is the decay rate.*

In addition, the inactive clusters which have no trend merged for some time could be regarded as the outliers. On the other hand, they possibly turn to the active clusters in the future and thus should not be erased directly. Therefore, we are inspired by (Cao et al, 2006) and define a safe low-bound for the inactive clusters accroding as follows,

$$\xi = \frac{\lambda^{t_{ex}+\mathcal{T}_{ck}} - 1}{\lambda^{\mathcal{T}_{ck}} - 1} \tag{6}$$

where $t_{ex}$ is the time span that the cluster exsit and $\mathcal{T}_{ck}$ is the minimum check period. Obviously, when $t_{ex} = 0$, i.e. the cluster has just been created, $\xi = 1$, while $\xi = \mu$ when $t_{ex}$ approaches infinity. Generally, the longer the trend cluster exist, the larger $\xi$ is supposed to be. Finally, we specify an initial time interval when the algorithm starts, during which the trend clusters will not be

**Algorithm 3** AntClu Algorithm

**Input:** Trajectory Stream $\mathcal{S}$, parameters: $(\epsilon, \mu, \beta, \delta, \lambda)$
**Output:** $TC(t)$
**for** each arriving point $p^{(t_i)}$ in $\mathcal{S}$ **do**
  Update the trend $TR_i$ according to Algorithm 1.
  Merge $TR_i$ to the closest active/inactive cluster with Algorithm 3.
  Update the corresponding Cluster Features of trend cluster.
  **if** $t_i \mod \mathcal{T}_{ck} == 0$ **then**
    // Periodical Check
    **for** each active cluster $TC_{ac}$ **do**
      **if** $TC_{ac}.\tau < \mu$ **then**
        delete $TC_{ac}$
      **end if**
    **end for**
    **for** each inactive clusters $TC_{in}$ **do**
      Compute the low-bound $\xi$ according to the current timestamp
      **if** $TC_{in}.\tau < \xi$ **then**
        delete $TC_{in}$
      **end if**
    **end for**
  **end if**
**end for**
**return** $TC(t) = \{TC_{ac}, TC_{in}\}$

decade in their activation. The pseudo-code of AntClu is given in Algorithm 3.

Additionaly, we specify a initial phase for our algorithm, in which the initial sub-trajectory are grouped by Algorithm 2, whereas the initial trend clusters will not be decade in that phase.

## 6 Experiment

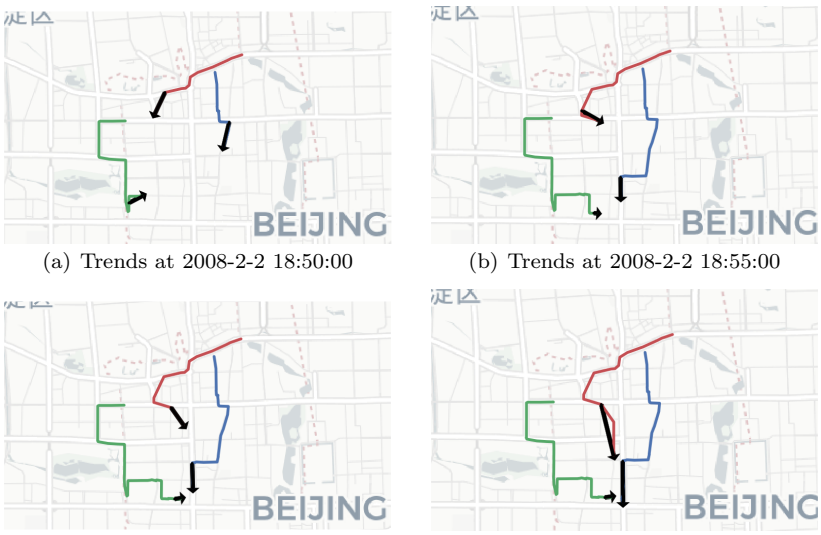
### 6.1 Experimental Setup

#### *6.1.1 Datasets*

In this paper, we use two real-life data sets, the track of taxis in Beijing and Shanghai, respectively, to evaluate the performance of AntClu. The first data set of taxi track is called T-drive, and was published in 2011 by MSRA, which contains the GPS trajectories of 10,357 taxis from 2008/2/2 to 2008/2/8 in Beijing. As for another data set SH-Taxi, it records the 24-hour track of 4316 taxis in 2007/2/20 in Shanghai. We reconstruct all the records by order of their timestamps. The experiments are performed on an Intel 3.6Hz PC with 8.0GB of RAM.

#### *6.1.2 Evaluation Measure*

To evaluate the effectiveness of our proposed method, we use Sum of Square Distance (SSQ) to measure the performance. Namely, $\sum_{i=1}^{N_c} \sum_{j=1}^{N_l} dist^2(L_j, TC_i.cen)$,




(a) Trends at 2008-2-2 18:50:00 (b) Trends at 2008-2-2 18:55:00

(c) Trends at 2008-2-2 19:00:00 (d) Trends at 2008-2-2 19:05:00

**Fig. 5** Dynamic representation of objects' trajectory over 3 trajectories in T-Drive. The black straight lines indicates the trends at each time point.

where the distance measurement is the euclidean distance between the center points of the line segments. A smaller SSQ indicates a better clustering quality. Meanwhile, we calculate the average processing time of each record to evaluate the efficiency of the algorithm.

## 6.2 Effectiveness

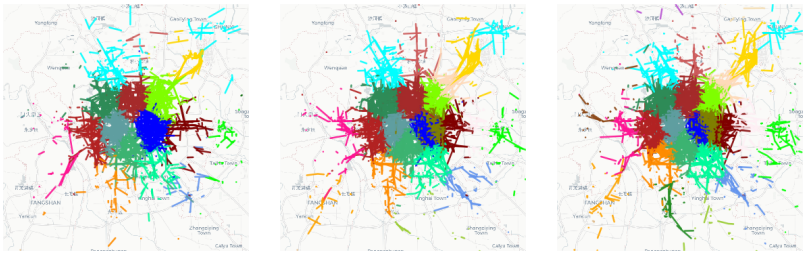
### *6.2.1 Dynamic Trend Representation*

To verify the effectiveness of the dynamic *trend* representation, we randomly pick up some tracks from T-Drive dataset and plot their sub-trajectories during three given time intervals. As illustrated in Fig. 5. We can see the trend representation is quite valid

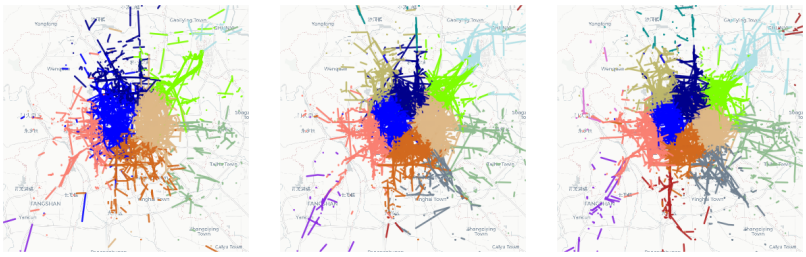
### *6.2.2 Comparison with TraClus Algorithm*

For better illustration of the advantage of AntClu over window-based methods, we compare the clustering results every 10 minutes, while the trajectory stream keeps coming through. We choose the widely used algorithm TraClus (Lee et al, 2007) as a baseline, which is also a density-based method. Specifically, for Traclus, we only group the sub-trajectory in a given time window instead the whole trajectory, in terms of the high computation cost of TraClus.

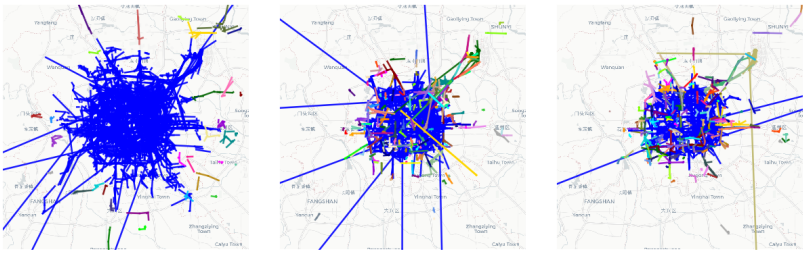
With a suitable parameter tuning, we set TraClus with $\epsilon = 0.0016$ and $minLine = 2$, which gives the best performance in following experiments.



(a) AntClu Clusters w.r.t. $\epsilon = 0.06$



(b) AntClu Clusters w.r.t. $\epsilon = 0.12$



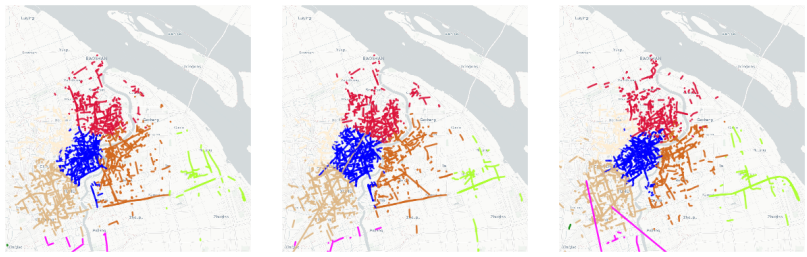
(c) TraClus Clusters

**Fig. 6** Clustering Results respectively at 18:40,18:50 and 19:00 in 2/2/2008 over T-Drive data. The different colors indicate different trend clusters at that moment. Specifically, Antclu processes the continuous trajectory data stream, whereas TraClus here groups the data during ten-minute time windows.

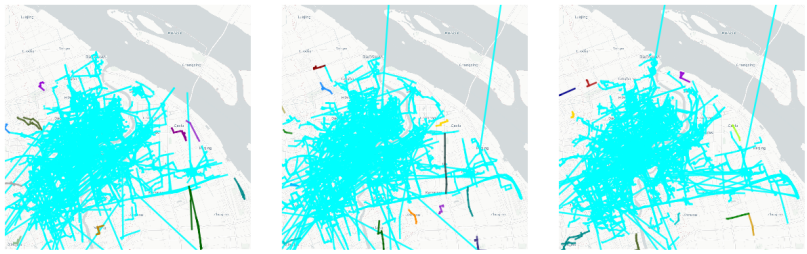
Meanwhile, for AntClu, we fix $\mu = 2, \delta = 0.004, \lambda = 0.8$ while varying the range parameter $\epsilon$(0.06 and 0.12). Notice, since the distance metrics used in AntClu and Traclus are totally different so that the compare between the values of $\epsilon$ is pointless.

As illustrated in Fig. 6 and Fig. 7, we can see that AntClu allows discovering credible groups of clusters and the macro-clustering output shows the significant evolution of the clusters. While the TraClus, grouping the whole-window span trajectories, can barely distinguish the variance of the clustering output between windows.

(a) AntClu Clusters w.r.t. $\epsilon = 0.06$



(b) AntClu Clusters w.r.t. $\epsilon = 0.12$



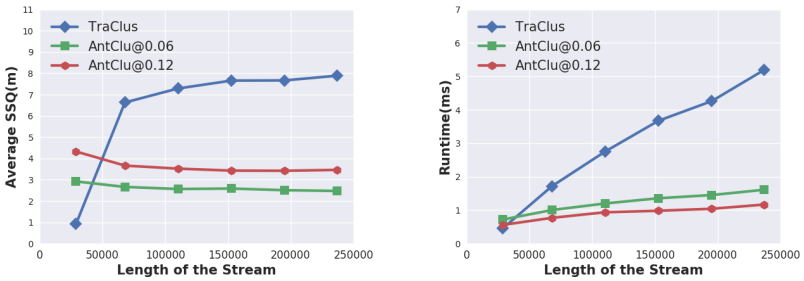
(c) TraClus Clusters

**Fig. 7** Clustering Results respectively at 12:10,12:20 and 12:30 in 2/20/2007 over SH-Taxi data. The different colors indicate different trend clusters at that moment. Specifically, Antclu processes the continuous trajectory data stream, whereas TraClus here groups the data duirng ten-minute time windows.

Moreover, Fig. 8 shows the average SSQ of AntClu comparing with TraClus. It further demonstrates that AntClu yields better clustering results compared to TraClus on both real-world data sets, and the advantage is more obvious with more points are involved.

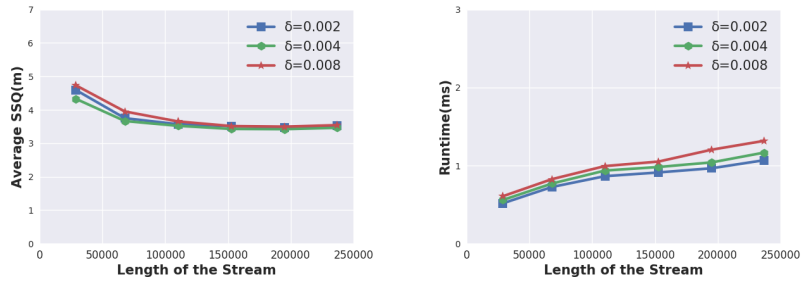
### 6.3 Efficiency

Here, we further compare the efficiency between AntClu and TraClus. Fig.8(b) plots the average running time with varying record number in the stream. We




(a) Average SSQ versus stream length (b) Average runtime versus stream length

**Fig. 8** The performance comparing AntClu with TraClus




(a) Average SSQ versus $\delta$ (b) Average runtime versus $\delta$

**Fig. 9** The performance w.r.t $\delta$

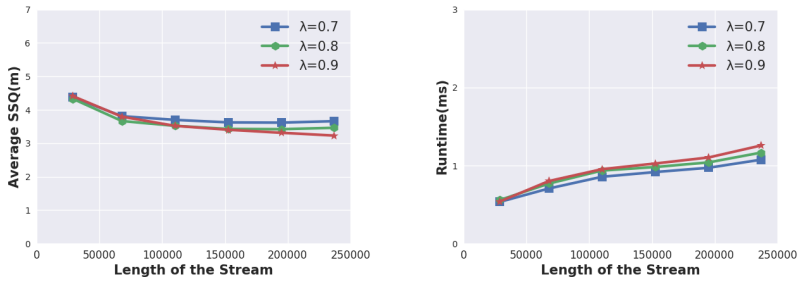
can observe that both algorithm is comparable, and AntClu shows a relatively better result w.r.t. $\epsilon = 0.06$ and 0.12.

### 6.4 Sensitivity Analysis

In this section, we perform sensitivity analysis for AntClu with respect to the error threshold $\delta$ and decay rate $\lambda$. From Fig. 6.4 and **??**, we can see that performance of AntClu is quite stable by varing the values of $\delta$ and *lambda*.

## 7 Conclusion

In this paper, we introduce a new trajectory clustering algorithm based on dynamic trend representation and density-based data stream clustering. Specifically, we propose a novel line segment distance measurement to find the representative line segments to represent the current trend. A data structure, called *trend*, is further proposed to manipulate incoming data, which is independent of time-window. Finally, the density-based data stream clustering




(a) Average SSQ versus $\lambda$ (b) Average runtime versus $\lambda$

**Fig. 10** The performance w.r.t $\lambda$

is adapted to trajectory clustering. Equipped with the dynamic trend representation and density-based trajectory clustering approach, AntClu shows its superiority over TraClus in terms of both effectiveness and efficiency on real-world data sets.

## References

Cao F, Ester M, Qian W, Zhou A (2006) Density-based clustering over an evolving data stream with noise. In: Siam International Conference on Data Mining, April 20-22, 2006, Bethesda, Md, Usa, pp 328–339

Chen N, Ribeiro B, Vieira A, Chen A (2013) Clustering and visualization of bankruptcy trajectory using self-organizing map. Expert Systems with Applications 40(1):385–393

Deng Z, Hu Y, Zhu M, Huang X, Du B (2015) A scalable and fast optics for clustering trajectory big data. Cluster Computing 18(2):549–562

Gudmundsson J, van Kreveld M (2006) Computing longest duration flocks in trajectory data. In: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems, ACM, pp 35–42

Han B, Liu L, Omiecinski E (2012) Neat: Road network aware trajectory clustering. In: IEEE International Conference on Distributed Computing Systems, pp 142–151

Han B, Liu L, Omiecinski E (2015) Road-network aware trajectory clustering: Integrating locality, flow, and density. IEEE Transactions on Mobile Computing 14(2):416–429

Hu W, Li X, Tian G, Maybank S, Zhang Z (2013a) An incremental dpmm-based method for trajectory clustering, modeling, and retrieval. IEEE Transactions on Pattern Analysis & Machine Intelligence 35(5):1051–65

Hu W, Li X, Tian G, Maybank S, Zhang Z (2013b) An incremental dpmm-based method for trajectory clustering, modeling, and retrieval. IEEE transactions on pattern analysis and machine intelligence 35(5):1051–1065

Jensen CS, Lin D, Ooi BC (2007) Continuous clustering of moving objects. IEEE Transactions on Knowledge & Data Engineering 19(9):1161–1174

Jeung H, Yiu ML, Zhou X, Jensen CS, Shen HT (2008) Discovery of convoys in trajectory databases. Proceedings of the VLDB Endowment 1(1):1068–1080

Lee JG, Han J, Whang KY (2007) Trajectory clustering:a partition-and-group framework. In: ACM SIGMOD International Conference on Management of Data, pp 593–604

Li Y, Han J, Yang J (2004) Clustering moving objects. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 617–622

Li Z, Ding B, Han J, Kays R (2010a) Swarm: Mining relaxed temporal moving object clusters. Proceedings of the VLDB Endowment 3(1-2):723–734

Li Z, Lee JG, Li X, Han J (2010b) Incremental clustering for trajectories. In: International Conference on Database Systems for Advanced Applications, pp 32–46

Mao J, Song Q, Jin C, Zhang Z, Zhou A (2016) TSCluWin: Trajectory Stream Clustering over Sliding Window. Springer International Publishing

Mohammed N, Fung BCM, Debbabi M (2009) Walking in the crowd:anonymizing trajectory data for pattern analysis. In: ACM Conference on Information and Knowledge Management, pp 1441–1444

Monreale A, Pinelli F, Trasarti R, Giannotti F (2009) Wherenext:a location predictor on trajectory pattern mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July, pp 637–646

Nanni M, Pedreschi D (2006) Time-focused clustering of trajectories of moving objects. Journal of Intelligent Information Systems 27(3):267–289

Phang TL, Neville MC, Rudolph M, Hunter L (2003) Trajectory clustering: a non-parametric method for grouping gene expression time courses, with applications to mammary development. Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing 123(2):351

Piciarelli C, Foresti GL (2006) On-line trajectory clustering for anomalous events detection. Pattern Recognition Letters 27(15):1835–1842

da Silva TLC, Zeitouni K, de Macêdo JA (2016) Online clustering of trajectory data stream. In: Mobile Data Management (MDM), 2016 17th IEEE International Conference on, IEEE, vol 1, pp 112–121

Silva TLCD, Zeitouni K, Macêdo JAFD, Casanova MA (2016) A framework for online mobility pattern discovery from trajectory data streams. In: IEEE International Conference on Mobile Data Management, pp 365–368

Tang LA, Zheng Y, Yuan J, Han J, Leung A, Hung CC, Peng WC (2012) On discovery of traveling companions from streaming trajectories. In: Data Engineering (ICDE), 2012 IEEE 28th International Conference on, IEEE, pp 186–197

Won JI, Kim SW, Baek JH, Lee J (2009) Trajectory clustering in road network environment. In: Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on, pp 299–305

Wu R, Luo G, Shao J, Tian L, Peng C (2018a) Location prediction on trajectory data: A review. Big Data Mining and Analytics 1(2):108–127

Wu R, Luo G, Yang Q, Shao J (2018b) Learning individual moving preference and social interaction for location prediction. IEEE Access 6:10675–10687

Yanwei, WANG, WANG, Xiaodong (2013) Continuous clustering trajectory stream of moving objects. China Communications 10(9):120–129

Zheng Y (2015) Trajectory Data Mining: An Overview. ACM

Zheng Y, Capra L, Wolfson O, Yang H (2014) Urban computing: Concepts, methodologies, and applications. Acm Transactions on Intelligent Systems & Technology 5(3):38