DOI: 10.2298/CSIS120723049Y

Online Clustering for Trajectory Data Stream of Moving Objects

Yanwei Yu[1,2], Qin Wang[1,2], Xiaodong Wang[1], Huan Wang[1], and Jie He[1]

[1] School of Computer and Communication Engineering,
University of Science and Technology Beijing
100083 Beijing, China
yuyanwei0530@gmail.com
[2] Beijing Key Laboratory of Knowledge Engineering for Materials Science
100083 Beijing, China
wangqin@ies.ustb.edu.cn

Abstract. Trajectory data streams contain huge amounts of data pertaining to the time and position of moving objects. It is crucial to extract useful information from this peculiar kind of data in many application scenarios, such as vehicle traffic management, large-scale tracking management and video surveillance. This paper proposes a density-based clustering algorithm for trajectory data stream called CTraStream. It contains two stages: trajectory line segment stream clustering and online trajectory cluster updating. CTraStream handles the trajectory data of moving objects as an incremental line segment stream. For line segment stream clustering, we present a distance measurement approach between line segments. Incremental line segments are processed quickly based on previous line clusters in order to achieve clustering line segment stream online, and line-segment-clusters in a time interval are obtained on the fly. For online trajectory cluster updating, TC-Tree, an index structure, which stores all closed trajectory clusters, is designed. According to the line-segment-cluster set, the current closed trajectory clusters are updated online based on TC-Tree by performing proposed update rules. The algorithm has exhibited many advantages, such as high scalability to process incremental trajectory data streams and the ability to discover trajectory clusters in data streams in real time. Our performance evaluation experiments conducted on a number of real and synthetic trajectory datasets illustrate the effectiveness, efficiency, and scalability of the algorithm.

Keywords: trajectory stream clustering, density-based clustering, line segment cluster, trajectory cluster, TC-Tree, online.

1. Introduction

Recently, advances in GPS, smartphones, and other electronic monitoring devices have facilitated the speed of collecting coordinates of moving objects. In these real time applications, the location data of mass moving objects are received in the form of trajectory streams. From this huge volumes of data, how

to extract useful information in the trajectory streams has become very important. Therefore, incremental trajectory data stream mining is receiving a lot of attentions. Trajectory clustering, which groups the trajectories of moving objects to reveal interesting correlations among them, has a number of applications in vehicle traffic management, logistics management, pattern recognition, and behavior analysis [21]. For example, clustering the trajectory data stream of moving objects in real time can obtain some motion patterns to help administrators to predict movement trends and prevent anomalous events from occurring in a large security system. Density-based clustering can discover the clusters with arbitrary shapes and filter out noise, which is very useful for identifying the relations of moving objects whose mobile direction is stochastic and unpredictable. Thus, an efficient clustering algorithm based on density is essential for trajectory data streams analysis in these applications with real time constraint.

In this paper, regarding to clustering trajectory data streams that pertain to time and the position of moving objects, we propose a clustering algorithm, CTraStream, which can discover the trajectory clusters with explicit temporal information and spatial information from data streams. The following contributions are made: First, we provide a definition of distance measure between trajectory line segments, and, by employing the distance measure, CTraStream performs density-based clustering on received trajectory line segment streams to obtain line segment clusters in real time. Second, a TC-Tree structure is designed, which stores the closed trajectory clusters and represents the relationship among trajectory clusters. Finally, the trajectory clusters on TC-Tree is updated according to the line segment clusters online. Comprehensive empirical studies on real data and large synthetic data demonstrate the clustering effectiveness, efficiency, and scalability of our developed algorithm.

The remainder of this paper is organized as follows: We discuss related clustering algorithms in Section 2. In Section 3, we first define the basic notions of our algorithm and introduce the general framework of CTraStream. A detailed description of CLnStream, a density-based line segment stream clustering algorithm, is provided in Section 4, and TraCluUpdate, a online update process of trajectory cluster, is presented in Section 5. Evaluation experiments of effectiveness and efficiency are shown in Section 6. Finally, Section 7 presents conclusions and points out directions for future work.

2. Related Works

Typical clustering methods based on density, such as algorithms based on local connectivity DBSCAN [18], OPTICS [16], and algorithms based on density function DENCLUE [1], give the notion of density. These algorithms can discover clusters with arbitrary shapes in spatial databases. However, they can't be applied to clustering trajectory data. Gaffney et al. [5][4] propose a clustering algorithm based on probabilistic model for trajectories. Gaffney's algorithm considers the trajectory as a whole and uses a mixture of regression models to represent a set of trajectories. Then, unsupervised learning is carried out

using the maximum likelihood principle. Specially, the EM algorithm is used to estimate hidden parameters involved in probability models, and then determine the clusters membership. But the basic processing unit of the clustering is the whole trajectory. Nanni [15] adapts two classical distance-based clustering methods (K-means and hierarchical agglomerative clustering) to trajectories. Then, Nanni et al [19] propose a time-focused trajectory clustering algorithm based on density. In Their paper, a simple notion of distance between trajectories is first defined, and then a new approach to trajectory clustering problems based on OPTICS, called temporal focusing, is presented, which aims to exploit the intrinsic semantics of temporal dimensions to improve the quality of trajectory clustering.

Piciarelli et al. [2] propose a trajectory clustering algorithm suited for online behavior analysis. The incremental trajectories are matched with previous clusters in space and grouped into the closest clusters, and the clusters are organized in a tree-like structure, which can be used to detect anomalous events. Lee et al. [8] present a trajectory clustering framework based on density, called TRACLUS, which is a partition-and-group framework for discovering common sub-trajectories in trajectory databases. In the partition phase, trajectories are partitioned into a set of quasi-linear segments using the minimum description length principle. In the group phase, all line segments are grouped using a density-based clustering method, and a representative trajectory for each cluster is determined. Based on TRACLUS, Lee et al. [9] succeedingly propose a feature generation framework TraClass for trajectory classification. It generates a hierarchy of features by partitioning trajectories and exploring region-based and trajectory-based clustering. Zhenhui Li et al. [27] propose a clustering framework for incremental trajectory called TCMM, which contains online micro-cluster maintenance and offline macro-cluster creation. The former is used to incrementally update the micro-clusters that store compact summaries of similar line segments to reflect the changes; the latter performs macro-clustering based on the set of micro-clusters when a user requests to see current clustering result. All of TRACLUS, TraClass and TCMM employ the density-based clustering method, but trajectories are represented as sequences of line segments without explicit temporal information.

Gudmundsson et al. [11] study the computational complexity and approximation strategies for a few motion patterns, such as flock pattern, which is defined as a group of at least m moving objects such that the objects always lay inside a circle of given radius during a period longer than a given threshold. A similar objective is pursued in [6], which emphasizes efficiency issues. Yifan Li et al. [26] propose an extension of micro-clustering to moving objects–moving micro-clustering (MMC)–which groups line segments of trajectories that lie within a bounding rectangle of given size in a certain time interval. However, MMC algorithm uses K-mean algorithm [17] as the generic algorithm, so it has some limitations in clustering with any shapes. Elnekave et al. [20] present an incremental clustering method for discovering evolving groups of moving objects similar to MMC. In the method, trajectories are represented as a list of min-

imal bounding boxes (MBBs), and a similarity between two trajectories based on MBBs is defined. With the definition, the method employs a trajectory-fitted version of the incremental K-mean algorithm for clustering moving objects. With similar objective, Jensen et al. [3] present a scheme that is capable of incrementally clustering moving objects. It employs a notion of object dissimilarity and clustering features, and uses a quality measure for incremental clusters to identify clusters that are not compact enough after certain insertions and deletions. Although the objectives of such algorithms are a somewhat different, they focus on the efficient discovery of static clusters at variable time snapshots.

Won et al [12] propose a trajectory clustering scheme for vehicles moving on road networks, which defines a trajectory as a sequence of road segments a moving object has passed by. The scheme modifies and adjusts FastMap and hierarchical clusters by measuring total length of matched road segments, and yields fairly accurate clustering results. However, the algorithm only can work on clustering objects in a spatial network. Kalnis et al. [13] provide a formal definition of moving clusters to discover a set of objects that move close to each other for a long time interval. During the time interval, some objects may leave or enter the cluster, but the portion of common objects should be higher than a predefined threshold in two continuous timestamps. Jeung et al. [10] define a convoy pattern, in which a set of objects that move together in a cluster with arbitrary shapes during at least k continuous timestamps. They propose three algorithms involving trajectory simplification techniques to query the convoy patterns from trajectory databases in a filter-refinement framework. However, these works cannot handle trajectory data streams efficiently since clusters are re-calculated from scratch every time. Efficient maintenance of such meta-trajectory clusters on streams requires a thorough analysis of the spatio-temporal properties of moving objects, which is a key issue addressed in our work.

3. Problem Formulation

3.1. Notion Definition

Trajectory data stream in which there are N moving objects is defined as follows: S =($P_1^0 P_2^0 \ldots P_i^0 \ldots P_n^0 P_1^1 P_2^1 \ldots P_i^1 \ldots P_n^1 \ldots P_1^j P_2^j \ldots P_i^j \ldots P_n^j \ldots P_1^k P_2^k \ldots P_i^k \ldots P_n^k \ldots$). Where, P_i^j is location of the i^{th} moving object at the j^{th} timestamp. Real time positions of moving objects at each sampling time interval are received in form of data stream.

Definition 1. (Trajectory line segment) For moving object O, its position is p^i when timestamp is t_i, and position is p^{i+1} when timestamp is t_{t+1}. Vector $\overrightarrow{p^i p^{i+1}}$ is called a trajectory line segment of the moving object O, also known as line segment.

As shown in Fig. 1, l_i and l_j, respectively, is a line segment of the moving objects i and j. Since the line segment is a vector consisting of two continuous sampling positions of a moving object, line segments have characteristic of strong timeliness and directionality.

Definition 2. (Line segments distance) For line segments of two moving objects during same time interval, the distance between l_i and l_j is defined as formula (1), which is composed of three components: start distance, center distance, and end distance.

$$dist(l_i, l_j) = \alpha * d_{start}(l_i, l_j) + \beta * d_{center}(l_i, l_j) + \gamma * d_{end}(l_i, l_j) \quad (1)$$

$d_{start}(l_i, l_j) = |p_i^s - p_j^s|$ is the distance between p_i^s the starting point of segment l_i and p_j^s the starting point of segment l_j, $d_{center}(l_i, l_j) = |a - b|$ is the distance between a the midpoint of the segment l_i and b the midpoint of the segment l_j, $d_{end}(l_i, l_j) = |p_i^e - p_j^e|$ is the distance between p_i^e the end point of l_i and p_j^e the end point of l_j. α, β, γ, respectively, stands for the weight of the start distance, the center distance and end distance.

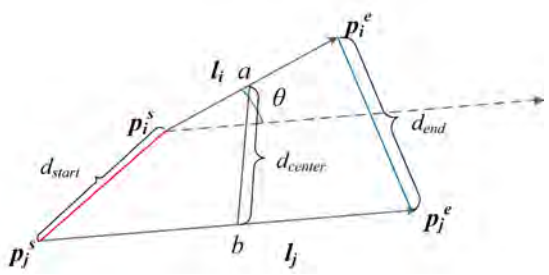


Fig. 1. Distance between line segments. θ is the included angle of l_i and l_j.

The weights are relative to θ the included angle of the two line segments, as defined in formula (2).

$$\begin{cases} \alpha = (1 - sin\theta/2)/3 \\ \beta = 1/3 \\ \gamma = (1 + sin\theta/2)/3 \end{cases} , 0 \leq \theta \leq \pi/2$$
$$\begin{cases} \alpha = (sin\theta/2)/3 \\ \beta = 1/3 \\ \gamma = (2 - sin\theta/2)/3 \end{cases} , \pi/2 < \theta \leq \pi \quad (2)$$

By formula (2), when the included angle θ is zero, the start distance, the center distance, and the end distance occupy the same proportion; while the weight of end distance increases in the distance formula as angle θ increases.

According to the definition of line segment distance, the line segments of numbers of moving objects during same time interval may distribute densely. Suppose D is the set of line segments during time interval [i, $i + 1$], e is the threshold of distance between two line segments.

Definition 3. (Neighborhood) For line segments l_i, the set $\{l|l \in D$ and $dist(l, l_i) \leq e\}$ is called the neighborhood of segment l_i, denoted by $NLns_D(l_i)$, and let$|NLns_D(l_i)|$ denotes the number of $NLns_D(l_i)$.

Definition 4. (Core line segment) A line segment l_i is a core line segment if $|NLns_D(l_i)|$ is not less than $MinLns$.

Definition 5. (Border line segment) If $|NLns_D(l_i)|$ is less than $MinLns$, but there is a core line segment l_j satisfying $l_j \in NLns_D(l_i)$, then the line segment l_i is called a border line segment of the line segment cluster containing l_j.

Definition 6. (Isolated line segment) A line segment l_i is an isolated line segment only if it is neither a core line segment nor a border line segment.

Definition 7. (Directly density-reachable) If a line segment l_i is the neighborhood of a core line segment l_j, then l_i is directly density-reachable from l_j.

Obviously, l_i is not necessarily a core line segment, so l_j is not necessarily directly density-reachable from l_i, therefore, directly density-reachable is asymmetrical.

Definition 8. (Density-reachable) A line segment l_i is density-reachable from a line segment l_j , if there is a series of line segments $l_i, l_{i+1}, \ldots, l_{j-1}, l_j \in D$ such that l_k directly density-reachable from $l_{k+1}(k = i, i+1, \ldots, j-1, j)$.

Like directly density-reachable, density-reachable is also asymmetrical, but the density-reachable is transitive. If l_i is density-reachable from a line segment l, and l is density -reachable from a line segment l_j, then l_i is density-reachable from l_j.

Definition 9. (Density-connected) If a line segment l_i and a line segment l_j are both density-reachable from a line segment l, then l_i is density-connected to l_j.

According to the definition 9, density-connected is symmetrical for a pair of line segments.

Definition 10. (Line-segment-cluster) Grouping the set D into a series of independent subset in which all line segments is density-connected to each other, each non-empty subset C is called a line-segment-cluster, abbr. l-s-cluster, if it satisfies the following two conditions:

(1) Connectivity: $\forall l_i; l_j \in C$, l_i is density-connected to l_j;

(2) Maximality: $\forall l_i; l_j \in D$, if $l_i \in C$ and l_j is density-reachable from l_i, then $l_j \in C$.

Each l-s-cluster can be comprised of at least one core line segment and all line segments being density-reachable from one of the core line segments. The set of line-segment-clusters in time interval $[i, i+1]$ is denoted as $LC_{[i,i+1]} = \{lc^1_{[i,i+1]}, lc^2_{[i,i+1]}, \ldots lc^k_{[i,i+1]}\}$.

Lemma 1. A core line segment must and only belongs to a l-s-cluster.

Proof: Let D be the current line segment set, and assume the core segment l belongs to two l-s-clusters $lc^1_{[i,i+1]}, lc^2_{[i,i+1]}$. Since the core line segment $l \in lc^1_{[i,i+1]}$, according to the definition of l-s-cluster, any line segment l' belonging to $lc^1_{[i,i+1]}$ is density-connected to l. Meanwhile, l is a core line segment of $lc^1_{[i,i+1]}$, the line segments which is density-connected to l are all density-reachable from

l, therefore, any line segment $l'(l' \in lc^1_{[i,i+1]})$ is density-reachable from l. Similarly, for any line segment $l''(l'' \in lc^2_{[i,i+1]})$ is also density-reachable from l. In conclusion, any line segment $l'(l' \in lc^1_{[i,i+1]})$ and $l''(l'' \in lc^2_{[i,i+1]})$ are all density-connected to each other, $lc^1_{[i,i+1]}, lc^2_{[i,i+1]}$ are the same l-s-cluster according to the definition of l-s-cluster. □

As shown in Fig. 2, there are 18 moving objects. Each line segment with an arrow stand for a trajectory line segment of the moving object, and the number marked on the line segment indicates the time interval of the line segment. Supposed $MinLns$ is 2, all line segments in a red ellipse constitute a line cluster.

Definition 11. (Trajectory cluster) For all moving objects in the trajectory stream N, a 2-tuples $(O, [i, j])$ is called a trajectory cluster if there exists $lc^{i_k}_{[k,k+1]} \in LC_{[k,k+1]}(k = i, i+1, \ldots, j-1)$ such that the set $O \subseteq \bigcap_{k=i}^{j-1} lc^{i_k}_{[k,k+1]}$ $(O \in N)$ during the time interval $[i, j](j > i)$. O is the set of objects in the trajectory cluster, and i and j is the start time and end time of the trajectory cluster respectively.

If $(O, [i, j])$ is a trajectory cluster, and any 2-tuples $(O', [i, j])$ $(O' \supset O)$ is not a trajectory cluster, then $(O, [i, j])$ is object-closed.

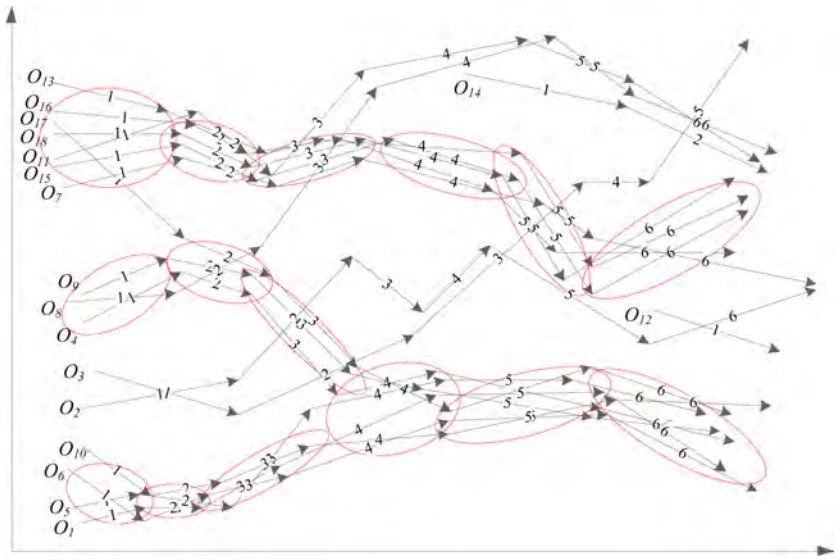


Fig. 2. An example of line segment clusters and trajectory clusters. Each line segment with an arrow represents a trajectory line segment of the moving object, and the number on the line segment indicates the time interval of the line segment.

Lemma 2. For a set $O(O \in N)$ including any m moving objects, if there exists $lc^{i_k}_{[k,k+1]} \in LC_{[k,k+1]}(k = i, i+1, \ldots, j-1)$ such that $O = \bigcap_{k=i}^{j-1} lc^{i_k}_{[k,k+1]}$

during time interval $[i,j](j > i)$, then the 2-tuples $(O,[i,j])$ is a object-closed trajectory cluster.

The lemma 2 is intuitive. Since $O = \bigcap_{k=i}^{j-1} lc_{[k,k+1]}^{i_k}$, for any $O'(O' \supset O)$,$O' \supset \bigcap_{k=i}^{j-1} lc_{[k,k+1]}^{i_k}$,so the 2-tuples $(O',[i,j])$ is not a trajectory cluster by the definition 11. Therefore, lemma 2 is proved.

If a 2-tuples $(O,[i,j])$ is a trajectory cluster, and $(O,[i-1,i])$ and $(O,[j,j+1])$ are not trajectory clusters, then $(O,[i,j])$ is time-closed.

A trajectory cluster $(O,[i,j])$ is called a closed trajectory cluster, if and only if $(O,[i,j])$ is both object-closed and time-closed. As shown in Fig. 2, the l-s-clusters of continuous time intervals constitute a closed trajectory cluster.

3.2. General Framework of CTraStream

CTraStream is divided into two online processes: line segment stream clustering, CLnStream, which clusters the line segment stream of moving objects online to obtain the l-s-clusters of current time interval, and the online update process of trajectory clusters, TraCluUpdate, which is responsible for online update trajectory clusters and extract closed trajectory clusters based on the TC-Tree structure. The framework of CTraStream algorithm is described in Algorithm 1, the two stages will be explained in section 4 and section 5, respectively.

Algorithm 1 CTraStream(Clustering Trajectory Stream)

Input : trajectory stream S; TC-Tree; L.
Parameter: e; *MinLns*; m.
Output : updated TC-Tree and closed trajectory clusters TC
Algorithm :
  /* Clustering for Line Stream *//*Section 4*/
1 : foreach(P_i^t in $S_{[t-1,t]}$)
2 : Execute CLnStream;// Algorithm2
3 : Get line clusters $LC_{[t-1,t]}$ from reslut;
  /* Trajectory Cluster Update *//*Section 5*/
4 : Execute TraCluUpdate($LC_{[t-1,t]}$, L,m);
  /* Query Closed Trajectory Clusters */
5: Report All *TC - Tree* Nodes;

4. CLnStream: Clustering Line Segment Stream based on Density

CLnStream performs clustering based on density for line segments of current time interval, and then gets l-s-clusters at the end of the current time interval. Compared with clustering results of sampling points, the l-s-clusters of moving objects represent the mobile status and trends more accurately.

When receiving a new point p_i^t , a new line segment $l_p = \overrightarrow{p^{t-1}p^t}$ is created and added to the line segments set D of the time interval $[t-1,t]$. The new line

segment only affects local l-s-clusters, so our algorithm only needs to perform the update processing on l_p and its neighborhood.

4.1. New Line Segment Affect

The new line segment l_p only effects local l-s-clusters of current time interval. The effected line segments could be divided into 2 groups:

New core line segments: with introduction of the new line segment l_p, some border line segments of its neighborhood may become a core line segment. And the clusters, in which these line segments are, will be updated. The set of new core line segments is denoted by $NewCoreLns_D(l_p)$, as shown in formula 3. If l_p is a core line segment, then $l_p \in NewCoreLns_D(l_p)$.

$$\begin{aligned} &NewCoreLns_D(l_p) = \{l||NLns_D(l)| \geq MinLns \\ &and\ |NLns_{D-l_p}| < MinLns\} \end{aligned} \tag{3}$$

Updated core line segments: Due to some border line segments become core line segments, these new core line segments will result in updating the l-s-clusters within their neighborhood according to the Lemma 1. So the previous core line segments in the neighborhood of $NewCoreLns_D(l_p)$ need to be updated as well. The set of core line segments which need to be updated is denoted by $UpdateCoreLns_D(l_p)$, as shown in formula 4.

$$\begin{aligned} &UpdateCoreLns_D(l_p) = \{l|\exists l_c \in NewCoreLns_D(l_p), l \in NLns_{D-l_p}(l_c) \\ &and\ |NLns_{D-l_p}(l)| \geq MinLns\} \end{aligned} \tag{4}$$

When adding the new line segment l_p, some density-connections in $D \times D$ may be established. And these connections bring an influence on the distribution of the l-s-clusters in which $NewCoreLns_D(l_p)$ and $UpdateCoreLns_D(l_p)$ are. According to $NewCoreLns_D(l_p)$ and $UpdateCoreLns_D(l_p)$, we can distinguish the following cases to process the l-s-clusters.

(1) Isolated line segment. If $NewCoreLns_D(l_p)$ is empty, then the new segment l_p is not a core segment, and there is no new core line segment. In this case, l_p is considered as an isolated line segment.

(2) Create l-s-cluster. If $NewCoreLns_D(l_p)$ is not empty and $UpdateCoreLns_D(l_p)$ is empty, then we can say there is no l-s-cluster in the local neighborhood of l_p. In this case, create new l-s-clusters according to the $NewCoreLns_D(l_p)$, and absorb line segments being density-reachable from the $NewCoreLns_D(l_p)$ into the new l-s-clusters.

(3) Extend l-s-clusters. If $NewCoreLns_D(l_p)$ is not empty and all elements of $UpdateCoreLns_D(l_p)$ belong to a l-s-cluster, then there is only one l-s-cluster around local neighborhood of the new segment l_p. In this case, the line segments being density-reachable from the core line segments are absorbed into the l-s-cluster according to the $NewCoreLns_D(l_p)$.

(4) Merge l-s-clusters. If $NewCoreLns_D(l_p)$ is not empty and the elements of $UpdateCoreLns_D(l_p)$ belong to several l-s-clusters, then there are more than

one l-s-clusters around the local neighborhood of the new line segment l_p. In this case, merge the l-s-clusters into one or several, and absorb the line segments being density-reachable from the $NewCoreLns_D(l_p)$ into the merged l-s-clusters.

4.2. Description of CLnStream

The pseudo code of CLnStream is shown as Algorithm 2. The detail steps of algorithm are described as following:

(1) Find the neighborhood of the new segment l_p.

According to section 4.1, find out the set $NewCoreLns_D(l_p)$ and $UpdateCoreLns_D(l_p)$.

(2) Clustering process if l_p is a core line segment, otherwise go to (3).

If l_p satisfies the conditions of core line segment, all elements in $NewCoreLns_D(l_p)$ are directly density-reachable from l_p, and all elements in $UpdateCoreLns_D(l_p)$ are also density-reachable from l_p. So the l-s-clusters in local area of the l_p should be merged into a l-s-cluster. Therefore, we first process the new line segment l_p if it is a core line segment, detail code is shown as line 3-13. If $UpdateCoreLns_D(l_p)$ is empty, then a new l-s-cluster is created, and all elements of $NewCoreLns_D(l_p)$ and all line segments being directly density-reachable from any element of $NewCoreLns_D(l_p)$ are absorbed into the new l-s-cluster. If the elements in the $UpdateCoreLns_D(l_p)$ only belong to one l-s-cluster, then $NewCoreLns_D(l_p)$ and all line segments being directly density-reachable from any element of $NewCoreLns_D(l_p)$ are absorbed into the cluster. If the elements of $UpdateCoreLns_D(l_p)$ are attributed to several l-s-clusters, then merging these clusters into a l-s-cluster, and absorbing $NewCoreLns_D(l_p)$ and line segments being directly density-reachable from any element in $NewCoreLns_D(l_p)$ into that cluster. In the pseudo code, $|UpdateCoreLns_D(l_p).clusters|$ stands for the number of l-s-clusters in $UpdateCoreLns_D(l_p)$, and $NLns_D(NewCoreLns_D(l_p))$ is the set of the neighborhood of all elements in the $NewCoreLns_D(l_p)$.

(3) Clustering process if l_p is not a core line segment.

If l_p does not satisfy the conditions of core line segment, there will exist multiple l-s-clusters within the neighborhood of l_p. Clustering processing is described as line 15-24. For every element of the $NewCoreLns_D(l_p)$ named l_q, we first find the core neighborhood of l_q denoted $CoreNLns_D(l_q)$, and then perform the clustering process under three cases. If the $CoreNLns_D(l_q)$ is empty, create a new l-s-cluster, and then absorb the l_q and line segments being directly density-reachable from l_q into the new cluster. If the elements in the $CoreNLns_D(l_q)$ only belong to one l-s-cluster, then l_q and all line segments being directly density-reachable from l_q are absorbed into the cluster. If elements of the $CoreNLns_D(l_q)$ are attributed to multiple l-s-clusters, then merge the multiple l-s-clusters into one l-s-cluster, and absorb the l_q and line segments being directly density-reachable from l_q into the merged l-s-cluster.

The l-s-clusters are obtained immediately when current time interval is over. After that, the TraCluUpdate conducts the update process for trajectory clusters on the set of l-s-clusters.

Algorithm 2 CLnStream(Clustering for Line Stream)

Input : new line segment l_p and existing line clusters
$LC_{[i-1,i]} = \{\{lc^1_{[i-1,i]}, lc^2_{[i-1,i]}, \cdots lc^k_{[i-1,i]}\}\}$
Parameter : e and $MinLns$
Output : updated $LC_{[i-1,i]}$ with l_p
Algorithm :
1 : find the $NewCoreLns_D(l_p)$;
2 : find the $UpdateCoreLns_D(l_p)$;
3 : *if* ($l_p \in NewCoreLns_D(l_p)$)
4 : *if* ($|\ UpdateCoreLns_D(l_p).clusters\ | = 0$)
5 : *new LineCluster* $lc \leftarrow \{NewCoreLns_D(l_p) \cup$
6 : $NLns_D(NewCoreLns_D(l_p))\}$;
7 : $LC_{[i-1,i]} \leftarrow LC_{[i-1,i]} \cup \{lc\}$;
8 : *else if* ($|\ UpdateCoreLns_D(l_p).clusters\ | = 1$)
9 : $lc^t_{[i-1,i]} \leftarrow lc^t_{[i-1,i]} \cup \{NewCoreLns_D(l_p) \cup$
10: $NLns_D(NewCoreLns_D(l_p))\}$;
11: *else if* ($(n = |\ UpdateCoreLns_D(l_p).clusters\ |) > 1$)
12: $lc^t_{[i-1,i]} \leftarrow lc^t_{[i-1,i]} \cup lc^{t_1}_{[i-1,i]} \cdots lc^{t_n}_{[i-1,i]} \cup$
13: $\{NewCoreLns_D(l_p) \cup NLns_D(NewCoreLns_D(l_p))\}$;
14:*else*
15: *foreach*(*Line* l_q in $NewCoreLns_D(l_p)$)
16: find the $NCoreLns_D(l_q)$;
17: *if* ($|\ NCoreLns_D(l_q).clusters\ | = 0$)
18: *new LineCluster* $lc \leftarrow \{l_q \cup NLns_D(l_q)\}$;
19: $LC_{[i-1,i]} \leftarrow LC_{[i-1,i]} \cup \{lc\}$;
20: *else if* ($|\ NCoreLns_D(l_q).clusters\ | = 1$)
21: $lc^t_{[i-1,i]} \leftarrow lc^t_{[i-1,i]} \cup \{l_q \cup NLns_D(l_q)\}$;
22: *else if* ($(n = NCoreLns_D(l_q).clusters\ |) > 1$)
23: $lc^t_{[i-1,i]} \leftarrow lc^t_{[i-1,i]} \cup lc^{t_1}_{[i-1,i]} \cdots lc^{t_n}_{[i-1,i]} \cup \{l_q \cup$
24: $NLns_D(l_q)\}$;

5. TraCluUpdate: online updating trajectory clusters on TC-Tree

Closed trajectory cluster is represented by a 2-tuples $(O, [i, j])$, which describes a maximum number of moving objects laying within a l-s-cluster in each time interval unit during the longest time interval $[i, j]$. This section introduces a online updating approach of closed trajectory clusters on TC-Tree storage structure.

5.1. TC-Tree

The TraCluUpdate maintains a single linked list L indexing the trajectory clusters that need to be updated and a closed trajectory clusters binary tree (TC-Tree). The structure of TC-Tree is shown in Fig. 3. TC-Tree is a binary tree with multiple root nodes, each non-root node represents a closed trajectory cluster, and left sub-tree of each root node stores a series of closed trajectory clusters with same start time. TC-Tree employs the “left child-right sibling” insert rule, so the relationship of all root nodes is sibling.

Supposed the set of l-s-clusters in current time interval $[j, j+1]$ is $LC_{[j,j+1]}$, for the node $TCNode = (O, [i, j])$, if its left child is not null, then its left child-node $(O', [i, j+1])$ satisfies the condition that $\exists lc^{i_k}_{[j,j+1]} \in LC_{[j,j+1]}$ makes $O' = O \cap lc^{i_k}_{[j,j+1]}$; if the right child of its left child-node is not null, then the right child-node of its left child-node $(O'', [i, j+1])$ satisfies the conditions of $O'' = O \cap lc^{i_k}_{[j,j+1]}$, and $O' \cap O'' = \phi$. If the left sub-tree of $TCNode$ is null, then for $\forall lc^{i_k}_{[j,j+1]} \in LC_{[j,j+1]}$, $(O \cap lc^{i_k}_{[j,j+1]}, [i, j+1])$ is not a closed trajectory cluster.

This description of TC-Tree further gives its following properties.

Property 1. For a node $(O, [start, end])$ of TC-Tree, any non-null node $(O', [start', end'])$ in its left sub-tree satisfies following three conditions: (1) $O' \subset O$; (2) $start' = start$; and (3) $end' > end$.

Property 2. For a node $(O, [start, end])$ of TC-Tree, any non-null node $(O', [start', end'])$ in its right sub-tree satisfies following two conditions: (1) $O' \cap O = \phi$; and (2) $start' = start$.

Fig. 3 shows an example of TC-Tree during time interval $[t1, t3]$. TC-Tree stores efficiently all closed trajectory clusters, and also reveals the spatio-temporal relationship of closed trajectory clusters. In addition, the L maintains the all current closed trajectory clusters of TC-Tree, shown as green linked list in Fig. 3.

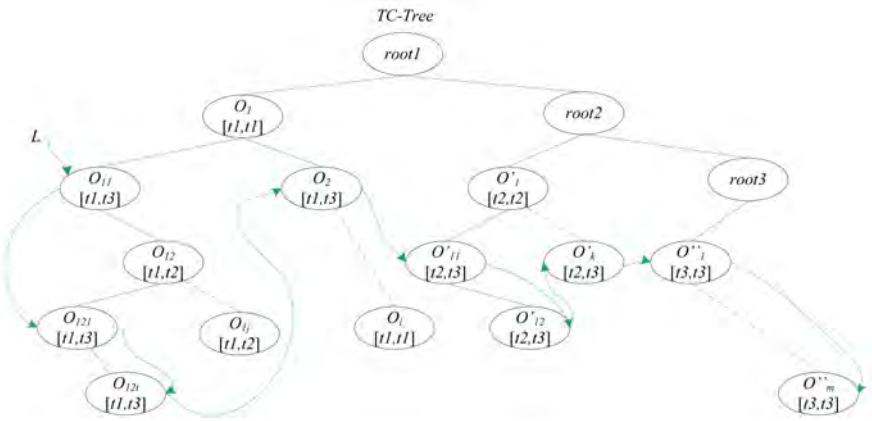


Fig. 3. An example of TC-Tree during time interval $[t1, t3]$.

5.2. Update Rules

Let $[j, j+1]$ be the current time interval, $LC_{[j,j+1]}$=$\{lc^1_{[j,j+1]}, lc^2_{[j,j+1]}, \ldots lc^k_{[j,j+1]}\}$ be the set of l-s-clusters obtained by CLnStream. The operation of inserting new nodes into TC-tree should follow the rule 1.

Rule 1. Consider inserting a new node into sub-tree of TC, a node of TC-tree, for left sub-tree inserting, if the left child of TC is null, then directly insert

the new node into its left child, otherwise insert the new node into the right sub-tree of its left child; for right sub-tree inserting, if the right child of TC is null, then directly insert the new node into its right child, otherwise insert the new node into the right sub-tree of its right child.

Lemma 3. Consider a node $(O, [start, j])$ of TC-Tree, if there exists a l-s-cluster $lc^i_{[j,j+1]}(1 \leq i \leq k)$ such that $lc^i_{[j,j+1]} \supseteq O$, then 2-tuples $(O, [start, j+1])$ is a current closed trajectory cluster.

Proof: the lemma 3 is intuitive. Since $(O, [start, j])$ is a closed trajectory cluster with respect to $[j-1, j]$, if there exists $lc^i_{[j,j+1]} \supseteq O$, then $O = O \cap lc^i_{[j,j+1]} = \bigcap_{k=start}^{j} lc^{i_x}_{[k,k+1]}$. Therefore, $(O, [start, j+1])$ is a current closed trajectory cluster. □

Rule 2. Consider a node $(O, [start, j])$ of L, if there exists a l-s-cluster $lc^i_{[j,j+1]}(1 \leq i \leq k)$ such that $lc^i_{[j,j+1]} \supseteq O$, then update this node to $(O, [start, j+1])$.

Lemma 4. Consider a node $(O, [start, j])$ of TC-Tree, if there exists a l-s-cluster $lc^i_{[j,j+1]}(1 \leq i \leq k)$ such that $|O \cap lc^i_{[j,j+1]}| \geq MinLns$ and $O \cap lc^i_{[j,j+1]} \neq O$, then the node $(O, [start, j])$ is a closed trajectory cluster, and 2-tuples $(O \cap lc^i_{[j,j+1]}, [start, j+1])$ is a object-closed trajectory cluster.

Proof: Due to there exists a l-s-cluster $lc^i_{[j,j+1]}$ such that $O \cap lc^i_{[j,j+1]} \neq O$, so $lc^i_{[j,j+1]} \neq O$ and $O \cap lc^i_{[j,j+1]} \subset O$. Therefore, $(O, [j, j+1])$ is not a trajectory cluster, further the node $(O, [start, j])$ is a closed trajectory cluster. And since$|O \cap lc^i_{[j,j+1]}| = |\bigcap_{k=start}^{j} lc^{i_k}_{[k,k+1]}| \geq MinLns$, it can be derived that $(O \cap lc^i_{[j,j+1]}, [start, j+1])$ is a object-closed trajectory cluster by Lemma 2. □

Lemma 5. (Backward closure checking) Consider a object-closed trajectory cluster $(O, [start, j+1])$, if there exists a current closed trajectory cluster $(O, [start', j+1])$ in TC-Tree such that $start' < start$, then $(O, [start, j+1])$ is not a closed trajectory cluster, otherwise $(O, [start, j+1])$ must be a current closed trajectory cluster.

Proof: First, since $(O, [start', j+1])$ is a closed trajectory cluster and $start' < start$, so $(O, [start', start])$ is a trajectory cluster, further we can get $(O, [start, j+1])$ is not a closed trajectory cluster.

Second, when there is no a current closed trajectory cluster $(O, [start', j+1])$ in TC-Tree such that $start' < start$, namely, $\forall start' < start$, $(O, [start', j+1])$ is not a closed trajectory cluster. we assume that $\exists (O, [start', start])(start' < start)$ is a trajectory cluster, due to $(O, [start, j+1])$ is a object-closed trajectory cluster, obviously, $(O, [start', j+1])$ also is a object-closed trajectory cluster. In another word, there must be a closed trajectory cluster $(O, [start'', j+1])$ such that $start'' \leq start'$. This is in conflict with the precondition, thus, the assumption is invalid. Therefore, when there is no a current closed trajectory cluster $(O, [start', j+1])$ in TC-Tree such that $start' < start$, $(O, [start, j+1])$ is a current closed trajectory cluster. □

L maintains all current closed trajectory clusters, in order to check whether a trajectory cluster is closed or not, so we only need to perform backward closure checking by comparing with nodes in L whose start time is earlier than $start$.

Rule 3. Consider every node $(O, [start, j])$ of L, if there exists a l-s-cluster $lc^i_{[j,j+1]}(1 \leq i \leq k)$ such that $|O \cap lc^i_{[j,j+1]}| \geq MinLns$ and $O \cap lc^i_{[j,j+1]} \neq O$, then backward check whether the $(O, [start, j+1])$ is closed or not, if yes, insert the trajectory cluster $(O, [start, j+1])$ into the left sub-tree of $(O, [start, j])$, and insert the $(O, [start, j+1])$ into L at next of $(O, [start, j])$. After $(O, [start, j])$ is updated, delete it from L.

By lemma 2, every 2-tuples $(lc^i_{[j,j+1]}, [j, j+1])(1 \leq i \leq k)$ is an object-closed trajectory cluster, hence we get rule (4).

Rule 4. For every l-s-cluster $lc^i_{[j,j+1]}(1 \leq i \leq k)$, if $(c^i_{[j,j+1]}, [j, j = 1])$ is a closed trajectory cluster, then insert the new node $(lc_{[j,j+1]^i}, [j, j+1])$ into left sub-tree of a new root node, and insert the new node into the tail of L.

By lemma 3 and lemma 4, all nodes indexed in L are current closed trajectory cluster after updating, and the start time of nodes is incremental from head to tail in L.

5.3. Algorithm of TraCluUpdate

Pseudo code of TraCluUpdate is shown in Algorithm 3. Line 1-16 describe the updating process of the linked list L, TraCluUpdate performs the rule 2 and rule 3 on nodes of L in turn first. Line 23-28 is detailed code of backward closure checking insert rule. Finally, rule 4 is carried out on L, as depicted line 17-22.

TC-Tree stores all closed trajectory clusters, we can traverse the TC-Tree to find out the closed trajectory clusters when user issues request to query. And because the updating linked list L maintains the all current closed trajectory clusters, we can directly search the L for responding the query on current closed trajectory clusters.

6. Experiment

In this section, comprehensive experiments are conducted on real data and synthetic datasets to evaluate the performance of CTraStream compared against CMC [10] and TCMM [27]. The recently and mostly related works to our algorithm are the convoy pattern discovery and incremental trajectory clustering. CMC proposed in [10] is a incremental solution for discovering convoy being similar with trajectory cluster from trajectory database. TCMM adapts the micro- and macro-clustering framework for handling incremental trajectory data. However, TCMM is not designed for clustering trajectory data streams. This is because sub-trajectory micro-clustering employed in TCMM has to wait for nontrivial number of new points accumulated to form sub-trajectories, which consumes additional buffer space and waiting time.

All the algorithms are implemented in C#, and all the experiments are performed on a 2.8 GHz I7 processor with 4GB memory. The system runs windows7 operating system and visual studio 2010.

Algorithm 3 TraCluUpdate(UpdateTrajectory clusters)

Input : line clusters $LC_{[j,j+1]}$; update linklist L;
minimal threshold parameter m.
Output : updated *TC-Tree* with $LC_{[j,j+1]}$
Algorithm :

```
1: new Node ← L.Head;
2: while(Node.next ≠ null)
3:    new TCNode = Node.next;
4:    new nextNode = TCNode.next;
5:    foreach(lc in LC[i-1,i])
6:       if(lc ⊇ TCNode.O)//rule2
7:          TCNode.end ← j+1;
8:       else if(|lc ∩ TCNode.O| ≥ m)//rule 3
9:          new N ← (lc ∩ TCNode.O, j+1);
10:         InsertTNode(TCNode,N);
11:         N.next = TCNode.next;
12:         TCNode.next = N;
13:   if(TCNode.lchild ≠ null)
14:      Node.next = TCNode.next;
15:      TCNode.next = null;
16:   Node = nextNode;
17:foreach(lc in LC[t-1,t])//rule 4
18:   new N ← (lc,[t - 1,t])
19:   if(BackwardCheck(null,N))
20:      insert N into TC-Tree.rchildTree; //insert rule
21:      Node.next = N;
22:      Node = Node.next;
   InsertNode(TCNode,N)
23:if(BackwardCheck(TCNode,N))
24:   insert N into TCNode.lchildTree; //insert rule
   BackwardCheck(TCNode.root,N)//backward check
25:foreach(pNode in (L.Head-TCNode))
26:   if(pNode.O == N.O)
27:      return false;
28:return true;
```

6.1. DataSets and Experimental Methodologies

Datasets. Data 1 is a real trajectory dataset from GeoLife project [25] conducted by Microsoft Research Asia, which includes GPS trajectory dataset of 178 users in a period of over four years (from April 2007 to October 2011). A GPS trajectory of this dataset is represented as a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. This dataset contains 17,621 trajectories with a total distance of 1,251,654 kilometers and a total duration of 48,203 hours. These trajectories were recorded by different GPS loggers and GPS-phones, and have a variety of sampling rates. 91 percent of the trajectories are logged in dense, e.g. every 1-5 seconds or every 5-10 meters per point.

Data 2 is a sub set of real trajectory data from T-Drive project [7] carried out by Microsoft Research Asia, which provides a smart driving direction services based on GPS trajectories of a large number of taxis. It helps users to find out the practically fastest path to a destination at a given departure time. It has been built based on a real-world trajectory dataset generated by 30,000 taxis

in Beijing in a period of 3 months, from February 2nd to February 8th in 2008. Data1 and data2 are also used in many previous studies, such as [24][14].

Data 3 is a GPS data set including 10 days' GPS data of 47 deer from Starkey project [23]. The dataset includes around 287,000 locations of elk, deer, and cattle that were collected during these seasons and years in the Main Study Area. Each animal location is provided as a point estimation (term as coordinations in the Easting and Northing direction) and also placed in the center point of each 30x30meter pixel that encompasses the point estimation. This database represents one of the largest data sets of animal locations ever compiled, released, and documented for use by scientists, students, and educators.

Data 4 is generated by Thomas Brinkhoff's network-based generator of moving objects [22], which includes trajectories of 200 moving objects with 200 sampling timestamps. This synthetic data generator has been used to generate synthetic data in many studies, also used in [13].

Experimental Methodologies. To confirm the behaviors of the algorithms in real applications, we run all the experiments utilizing both the synthetic and the real datasets. For the experiments, we run the experiments for multiple rounds, and get the average value.

We measure two key metrics for stream clustering algorithms, effectiveness and efficiency. The effectiveness of clustering algorithms refers to how the algorithm can effectively discover all clusters, while the efficiency stands for the clustering time, which is more important for stream clustering algorithms. In particular, we measure the average clustering time of each line segment and average running time it takes to clustering for a long time under different parameters. This running time includes the time utilized by the two stages of trajectory data stream clustering. For the effectiveness evaluation, we compare the clusters discovered by our method with CMC. We further verify the effectiveness of our algorithms with respect to different timestamps on real and synthetic datasets.

In addition, the effect of input parameters on efficiency is evaluated in parameter sensitivity analysis (section 6.4). Sensitivity analysis is the study of how the inputs effect the algorithm. The purpose is test the robustness of our algorithm in the presence of uncertainty. The experimental methodologies are also used for evaluating performance of clustering algorithms in [8][27].

6.2. Effectiveness

First, we verify the effectiveness of CTraStream on four datasets. Some tested cases of Clustering results of CTraStream, comparing with CMC, is shown in Fig. 4-Fig. 7. Setting same parameters for the two algorithms: e=150, $MinLns$=3 on data 1 and data 2, e=200, $MinLns$=3 on data 3, e=20, $MinLns$=3 on data 4. We check the results at two timestamps on each dataset. The experiment results show that most of clusters discovered by CTraStream are consist with convoys mined by CMC. In particular on data 3, when t=90, CTraStream discovers 44 closed trajectory clusters including 3 current closed trajectory clusters,

while CMC finds out 51 closed Convoys including 3 current closed convoys. When t=110, CTraStream discovers 50 closed trajectory clusters including 4 current closed trajectory clusters, while CMC finds out 61 closed convoys including 4 current closed convoys. As shown in Fig. 4, two closed convoys mined by CMC on data 1 lose a closely moving object compared with the corresponding trajectory clusters discovered by CTraStream. And Fig. 5(a) and Fig. 5(b) demonstrate the same situation on data 2. Fig. 6(a) and 7(a) show two current closed trajectory clusters on data 3 and data 4 respectively, and Fig. 6(b) and 6(b) show corresponding current closed convoys. The trajectory clusters discovered by CTraStream contain more relative time information compared with the convoys discovered by CMC. Fig. 6(c) and 6(d) depict respectively a closed trajectory cluster and two closed convoys on data 3. Obviously, CMC divides a group movement into two convoys. In conclusion, compared with CMC, the trajectory clusters discovered by CTraStream are closer to real movement track of moving objects with respect to the same parameters.

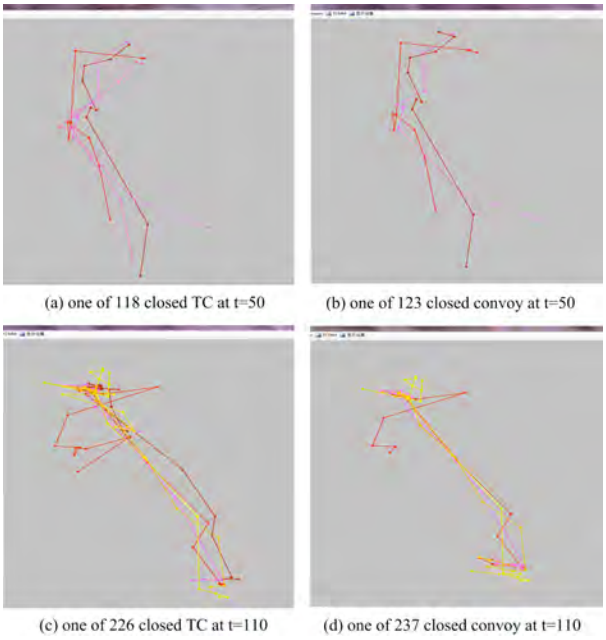


Fig. 4. Effectiveness comparison between CTraStream and CMC on data 1

6.3. Efficiency

Scalability. Next, a running time test is carried out on four datasets to verify the data stream processing efficiency of CTraStream with parameters e=150,

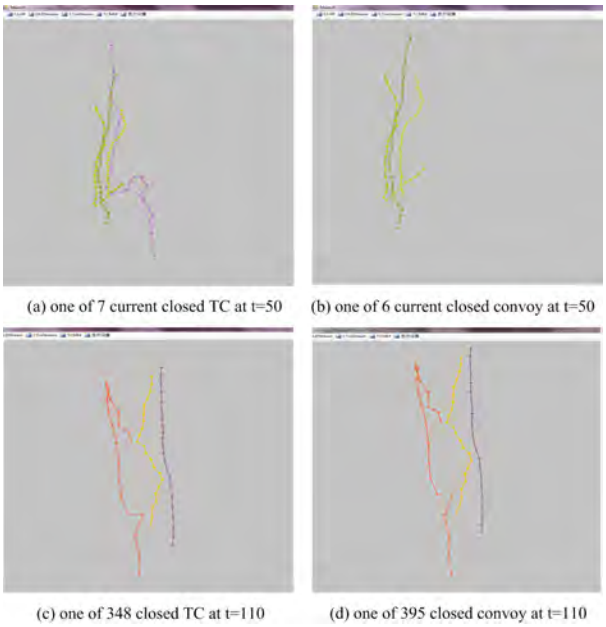


Fig. 5. Effectiveness comparison between CTraStream and CMC on data 2

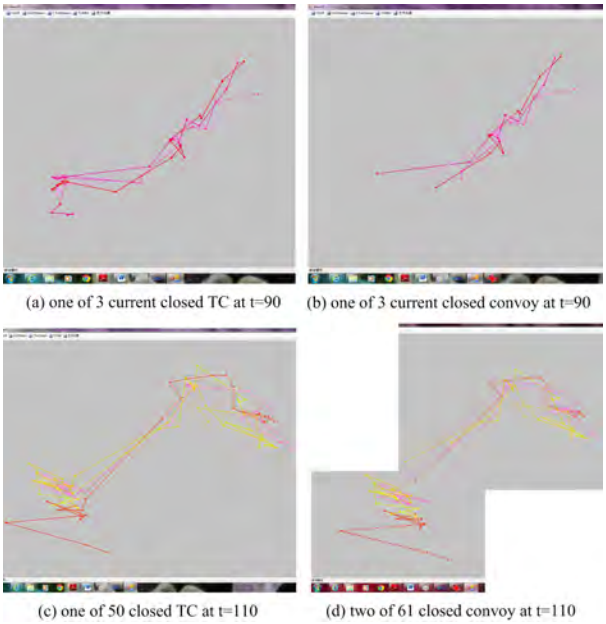


Fig. 6. Effectiveness comparison between CTraStream and CMC on data 3

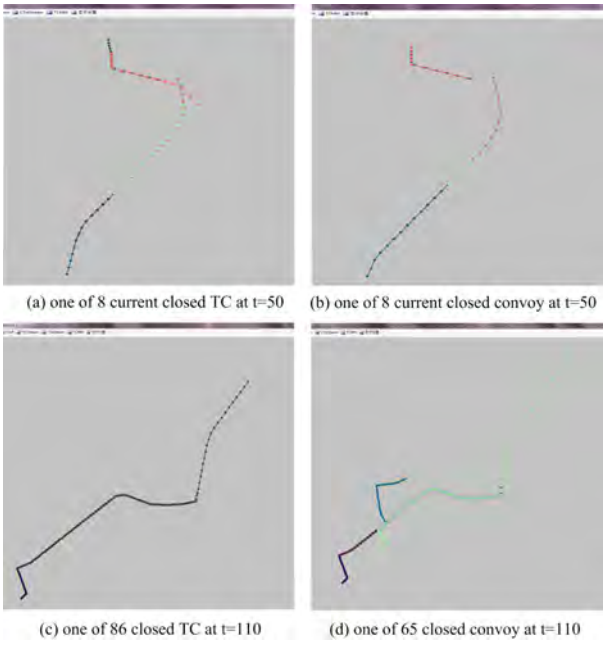


Fig. 7. Effectiveness comparison between CTraStream and CMC on data 4

$MinLns$=3 on data1 and data 2, e=150, $MinLns$=3 on data 3, e=40, $MinLns$=4 on data 4. The running time of CTraStream in each time interval unit is reported in Fig. 8, in which the running time on data 1, data 3 and data 4 refer to left y-axis, and the running time on data 2 refers to right y-axis. The results show that the running time in every time interval unit fluctuates smoothly within a small range with increase or decrease in timestamp. And the results also demonstrate that our algorithm have a good scalability. In addition, Statistics on the experimental results shows that average clustering time of CTraStream for a line segment is about 0.42 ms, 0.64 ms, 0.42 ms and 0.3 ms on data 1, data 2, data 3 and data 4, respectively.

Efficiency comparison. With increase in received trajectory points, the running time of CTraStream compared against CMC, TCMM on data 1-4 is presented in Fig. 9. It is noted that CTraStream and CMC use same parameter values on same datasets, while TCMM sets d_{max}=4*e on the corresponding datasets. The running time is the cumulative time of clustering all received trajectories from beginning. In Fig. 6, CtraStream and CMC refer to the left y-axis, and TCMM refers to right y-axis. The results of our experiments indicate that the cumulative time of CTraStream is slightly linear over the number of received trajectory points. Although TCMM can deal with the incremental trajectory data, it only considers the spatial correlation. And since treating each isolated line segment as a micro-cluster, hence the efficiency will drop with increase of re-

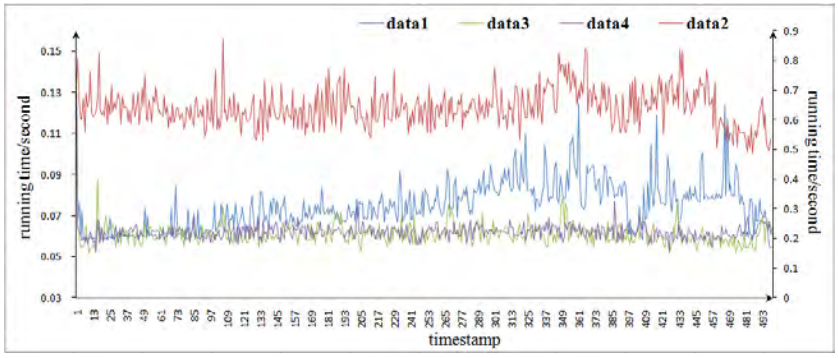


Fig. 8. Clustering time of CTraStream during each time interval unit on four datasets

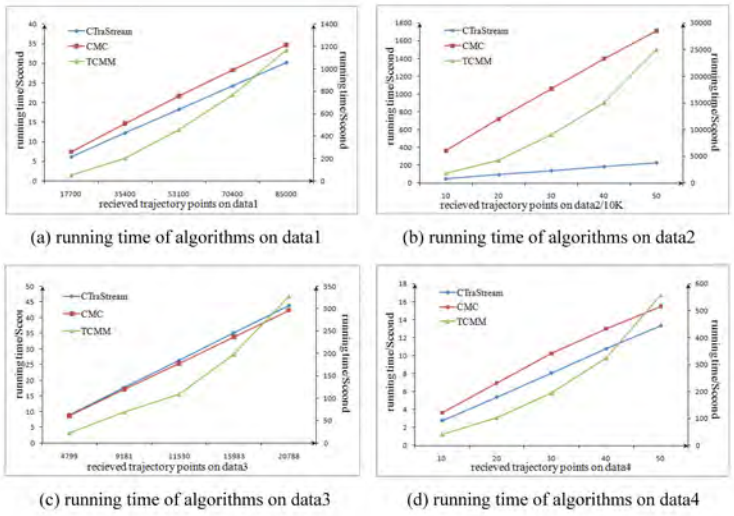
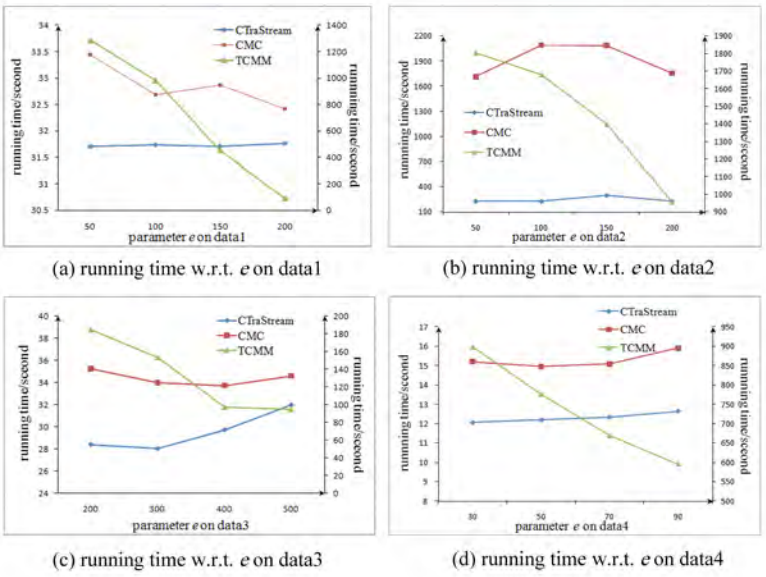


Fig. 9. Comparisons of clustering time of CTraStream, CMC, and TCMM on four datasets

ceived trajectory points. CMC employs DBSCAN to cluster received points on every timestamp, and then extracts their common objects and finds the maximum timestamp set to form convoys. When the number of moving objects is bigger, CMC will have a much lower efficiency than CTraStream, as shown in the experiment on data 2.



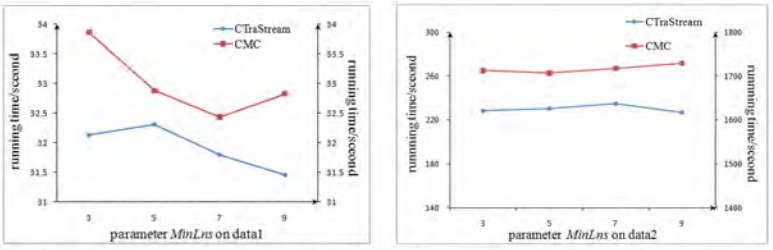
(a) running time w.r.t. e on data1 (b) running time w.r.t. e on data2

(c) running time w.r.t. e on data3 (d) running time w.r.t. e on data4

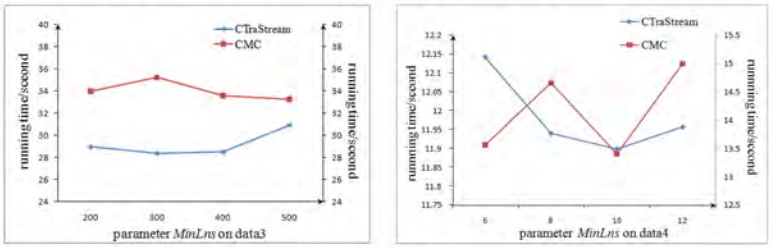
Fig. 10. Efficiency of CTraStream, CMC and TCMM w.r.t. e on four datasets

6.4. Parameter Sensitivity

The step of line segment stream clustering in CTraStream has two important parameters: e and $MinLns$. They determine the neighborhood range of the line segments. A larger e and a smaller $MinLns$ results in large l-s-clusters during time interval unit, so that the trajectory clusters is more relax; while a small e and a larger $MinLns$ has an opposite effect. We evaluate the effect of the parameters on the clustering efficiency on four datasets for CTraStream, TCMM and CMC. Fig. 10 shows the results with respect to e, CTraStream and CMC refer to the left y-axis, and TCMM refers to the right y-axis. We can see that with increase of e, CTraStream has a slight growth in running time, and for CMC, there is a slight fluctuation, but it has larger change range compared with CTraStream, while TCMM have a significant drop in running time. TCMM only has one parameter of e, so only the running time of CTraStream and CMC with respect to $MinLns$ are shown in Fig. 11. CTraStream refers to the left y-axis, and CMC refers to the right y-axis. Variation of $MinLns$ has a modest effect on efficiency of CTraStream and CMC. However, comparing with TCMM and CMC, the time taken by CTraStream is still significantly less specially for high volumes of moving objects.



(a) running time w.r.t. *MinLns* on data1 (b) running time w.r.t. *MinLns* on data2



(c) running time w.r.t. *MinLns* on data3 (d) running time w.r.t. *MinLns* on data4

Fig. 11. Efficiency of CTraStream and CMC w.r.t. $MinLns$ on four datasets

7. Conclusion

Discovering trajectory clusters from trajectory data stream is challenge, and existing solutions to related problems are ineffective in processing data stream. This paper proposes a trajectory data stream clustering algorithm based on density, CTraStream, to extract all closed trajectory clusters from trajectory data streams online. CTraStream considers the incremental trajectories of mass moving objects as line segment streams. In the first stage, a density based line segment stream clustering algorithm is presented and a set of l-s-clusters is obtained at end of current time interval. In the second stage, a TC-tree is maintained to store all closed trajectory clusters, an is updated according to the set of l-s-clusters by performing the update rules.

CTraStream effectively solves the problem of trajectory data stream clustering. Our experimental comparison of CTraStream, CMC and TCMM on a number of real and synthetic data sets demonstrates the effectiveness and efficiency of our algorithm. However, there may be problems where this assumption does not hold, i.e. parameters can adjust adaptively with change of trajectory data stream. In future work, we plan to investigate this case and apply CTraStream into real-time monitoring system of mass moving objects to detect automatically anomalous motion patterns.

Acknowledgments. This work is supported by National Natural Science Foundation of China (No. 61172049, 61003251), National High Technology Research and Development Program of China (863 Program)(No. 2011AA040101), Doctoral Fund of Ministry of Education of China (No. 20100006110015), Project of Beijing Municipal Science and Technology Commission (Z111100054011078), the 2012 Ladder Plan Project of Beijing Key Laboratory of Knowledge Engineering for Materials Science (No. Z121101002812005), and construction project of Beijing Municipal Commission of Education.

References

1. Alexander Hinneburg, D.A.K.: An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining. pp. 58–65. AAAI Press, New York, USA (1998)
2. C. Piciarelli, G.F.: On-line trajectory clustering for anomalous events detection. Pattern Recognition Letters 27(15), 1835–1842 (2006)
3. Christian S. Jensen, Dan Lin, B.C.O.: Continuous clustering of moving objects. IEEE Transactions on Knowledge and Data Engineering 19(9), 1161–1174 (2007)
4. Gaffney, S., R.A.S.P.C.S., Ghil, M.: Probabilistic clustering of extratropical cyclones using regression mixture models. Climate Dynamics 29(4), 423–440 (2007)
5. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining. pp. 63–72. ACM Press, San Diego, California (1999)
6. Hwang, S.-Y., L.Y.H.C.J.K..L.E.P.: Mining mobile group patterns: A trajectory-based approach. In: Proceedings of 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 713–718. Springer-Verlag, Berlin, Heideberg (2005)
7. J., Yuan, Y.Z.C.Z.W.X.X.X.G.S.Y.H.: T-drive: driving directions based on taxi trajectories. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 99–108. ACM Press, New York, USA (2010)
8. Jae-Gil Lee, J.H.: Trajectory clustering: A partition and group framework. In: Proceedings of ACM SIGMOD 2007 International Conference on Management of Data. pp. 593–604. ACM press, Beijing, China (2007)
9. Jae-Gil Lee, Jiawei Han, X.L.H.G.: Traclass: Trajectory classiffication using hierarchical region-based and trajectory-based clustering. Proceedings of the PVLDB Endowment 1(1), 1081–1094 (2008)
10. Jeungy H, Yiu Man Lung, Z.X.J.C.S.e.a.: Discovery of convoys in trajectory databases. Proceedings of the VLDB Endowment 1(1), 1068–1080 (2008)
11. Joachim Gudmundsson, Marc van Kreveld, B.S.: Efficient detection of motion patterns in spatio-temporal data sets. In: Proceedings of the 13th International Symposium of ACM Geographic Information Systems. pp. 250–257. ACM Press, New York, USA (2004)
12. Jung-Im Won, Sang-Wook Kim, J.H.B.J.L.: Trajectory clustering in road network environment. In: Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, 2009. pp. 299–305. IEEE Press, Nashville, TN (2008)
13. Kalnis P., Mamoulis N., B.S.: On discovering moving clusters in spatio-temporal data. In: Proceedings of the 9th international conference on Advances in Spatial and Temporal Databases. pp. 364–381. Springer-Verlag, Berlin, Heidelberg (2005)

14. Lu-An Tang, Yu Zheng, J.Y.J.H.: On discovery of traveling companions from streaming trajectories. In: Proceedings of IEEE 28th International Conference on Data Engineering, 2012. pp. 186–197. IEEE Press, Washington, USA (2012)
15. M., N.: Clustering methods for spatio-temporal data. Univ. of Pisa, Italy (2002), phD thesis
16. M. Ankerst, M. Breunig, H.P.K., J, S.: Optics: Ordering points to identify the clustering structure. In: Proceedings of ACM SIGMOD 1999 International Conference on Management of Data. pp. 49–60. ACM press, Pennsylvania, USA (1999)
17. MacQueen, J.: Some methods for classi?cation and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297. University of California Press, Berkeley, California (1967)
18. Martin Ester, Hans-Peter Kriegel, J.S.e.a.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Discovery and Data Mining. pp. 226–231. AAAI press, Portland, OR (1996)
19. Nanni M., P.D.: Time-focused clustering of trajectories of moving objects. Journal of Intelligent Information Systems 27(3), 267–289 (2006)
20. Sigal Elnekave, Mark Last, O.M.: Incremental clustering of mobile objects. In: Proceedings of IEEE 23rd International Conference on Data Engineering Workshop. pp. 585–592. IEEE Press, DC, USA (2007)
21. Slava Kisilevich, Florian Mansmann, M.N.S.R.: Spatio-temporal clustering: a survey. In: Oded Z. Maimon, L.R. (ed.) Data Mining and Knowledge Discovery Handbook, LLC, vol. 6, pp. 855–874. New York : Springer (2010)
22. T., B.: Network- based generator of moving object. Institute of Angewandte Photogrammetric and Geoinformatik (2005), [Online]. Available: http://iapg.jade-hs.de-/personen/brinkhoff/generator/
23. USDA: The starkey project. Oregon Department of Fish and Wildlife, US (1996), [Online]. Available: http://www.fs.fed.us/pnw/starkey/
24. Wei Liu, Yu Zheng, S.C.J.Y., Xie, X.: Discovering spatio-temporal causal interactions in traffic data streams. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011. pp. 1010–1018. ACM Press, New York, USA (2011)
25. Y., Zheng, Y.C.X.X.W.Y.M.: Geolife2.0: A location-based social networking service. In: Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware. pp. 357–358. IEEE Press, Washington, USA (2009)
26. Yifan Li, Jiawei Han, J.Y.: Clustering moving objects. In: Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining. pp. 617–622. ACM Press, Seattle, Washington (2004)
27. Zhenhui Li, Jae-Gil Lee, X.L.e.a.: Incremental clustering for trajectories. In: Proceedings of 2010 International Conference on Database System For Advanced Applications. pp. 32–46. Springer LNSS, Tsukuba, Japan (2010)

Yanwei Yu, PhD candidate at the School of Computer and Communication Engineering, University of Science and Technology Beijing. His research interest covers spatio-temporal data mining and wireless sensor networks.

Qin Wang, Professor at the School of Computer and Communication Engineering, University of Science and Technology Beijing. Her research interest covers wireless sensor networks, wireless localization, data mining, and computer architecture.

Xiaodong Wang, Master student of University of Science and Technology Beijing. His main research interest is trajectory data mining.

Huan Wang, Undergrad student of University of Science and Technology Beijing. His main research interest is trajectory data mining.

Jie He, PhD, faculty at the School of Computer and Communication Engineering, University of Science and Technology Beijing. His research interest covers indoor positioning techniques and wireless sensor networks.

Received: July 23, 2012; Accepted: December 6, 2012.