Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 7523138, 9 pages
https://doi.org/10.1155/2017/7523138

Research Article

CC_TRS: Continuous Clustering of Trajectory Stream Data Based on Micro Cluster Life

Musaab Riyadh, Norwati Mustapha, Md. Nasir Sulaiman, and Nurfadhlina Binti Mohd Sharef

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor, Malaysia

Correspondence should be addressed to Musaab Riyadh; m.shibani1968@gmail.com

Received 24 February 2017; Accepted 18 April 2017; Published 20 July 2017

Academic Editor: Nazrul Islam

The rapid spreading of positioning devices leads to the generation of massive spatiotemporal trajectories data. In some scenarios, spatiotemporal data are received in stream manner. Clustering of stream data is beneficial for different applications such as traffic management and weather forecasting. In this article, an algorithm for Continuous Clustering of Trajectory Stream Data Based on Micro Cluster Life is proposed. The algorithm consists of two phases. There is the online phase where temporal micro clusters are used to store summarized spatiotemporal information for each group of similar segments. The clustering task in online phase is based on temporal micro cluster lifetime instead of time window technique which divides stream data into time bins and clusters each bin separately. For offline phase, a density based clustering approach is used to generate macro clusters depending on temporal micro clusters. The evaluation of the proposed algorithm on real data sets shows the efficiency and the effectiveness of the proposed algorithm and proved it is efficient alternative to time window technique.

1. Introduction

Recently, moving objects such as vehicles and animals are equipped with GPS devices; these devices leave digital traces (latitude, longitude) position at each moment. The cheap price of GPS devices leads to an exponential growth of trajectories data. Analysis of trajectory data leads to extraction curial information which helps the researchers to find solutions for many challenges such as traffic congestion [1]. One of the most important analysis tools is clustering; clustering aims to aggregate data in clusters such that the similarity among cluster members is high and the similarity of members belonging to different clusters is very low [2, 3]. Clustering of stream data is more complex than classical data, since clustering stream data faces a set of challenges: (i) single pass processing due to continuous arriving of data, (ii) unbounded size of data stream and limited memory space and time, and (iii) evolving data where the model underlying the data stream may change over time. Thus the clustering algorithm should be able to detect such changes [3, 4]. Many algorithms of data stream clustering depend on object based paradigm which consists of two phases: online phase and offline phase. The online phase stores summarized information of data stream in specific micro clusters which act as representative for raw data stream. When the size of micro clusters exceeds memory limitation, similar micro cluster will merge to reduce memory size. The offline phase which is evoked on user demand and density base clustering approach is used to cluster representative line of micro clusters to demonstrate the current results of stream data.

Problem Statement. Many existing algorithms such as TCMM and ConTraClu exploit time window technique to incrementally cluster trajectory data stream. Time window technique partitions trajectory stream data into equal temporal periods (time bins or time stamp) and clusters each period separately as illustrated in Figure 1. Starting clustering from scratch in each time bin leads to the following. (i) Disturbance occurs in clustering quality which centralizes in the border area between two adjacent time bins specially if it is very dense of trajectory segments since clustering process in time window technique creates new micro clusters (MC_n) for some segments S_i at the start of each timebin_{k+1} even though these segments are very close (within distance threshold) to

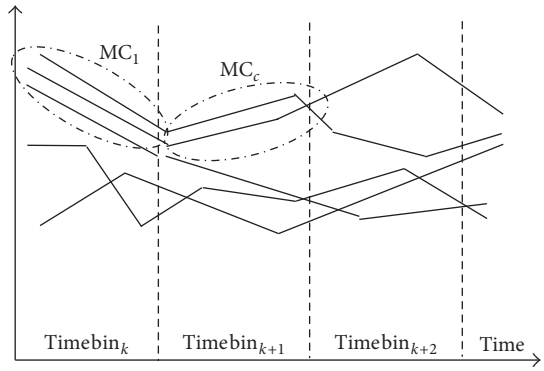


Figure 1: Time windows technique.

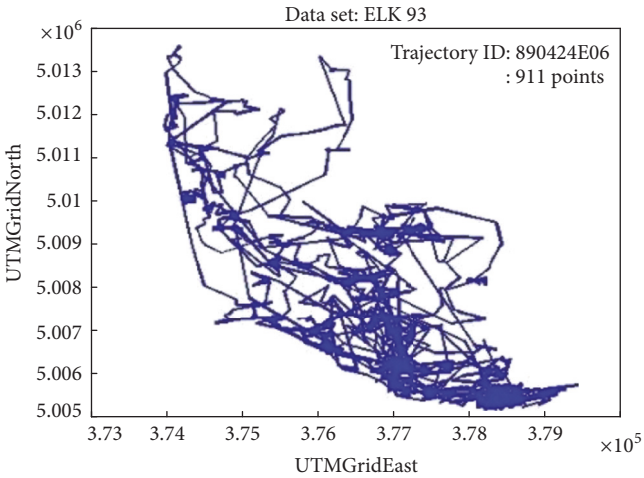


Figure 2: Trajectory of free moving object.

micro clusters (MC$_m$) at the end of previous timebin$_k$. (ii) It is true that TCMM algorithm combines some of MC$_n$ and MC$_m$ during merging stage to reduce memory space but that will be time consuming. In addition to these problems, TCMM framework merges similar micro cluster when its size exceeds a given memory space, and the merging process does not maintain temporal information and that is inconsistent with the complexity of free moving object since it visits the same spatial area many times in different periods of time as illustrated in Figure 2.

In this article, Continuous Clustering of Trajectory Stream Data Based on Micro Cluster Life (CC_TRS) algorithm is proposed; the algorithm assigns a life time for each newly created micro cluster, and the new upcoming segments will affect only the nonexpired clusters; for example, any segments in current time (dash line in Figure 3) will affect temporal micro cluster (B and C) and ignore A since it is expired. A Micro Cluster Life is a continuous clustering technique; therefore there is no need to divide stream data to time bins and start clustering from scratch as in time window technique. CC_TRS algorithm consists of two-phase micro clustering (online phase) and macro clustering (offline phase). In online phase, temporal micro cluster (TMC) data structure is defined to store summarized information for each group of

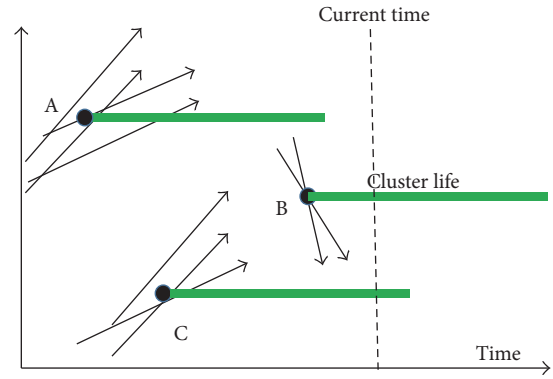


Figure 3: Cluster life technique.

similar segments; TMC is similar to micro cluster data structure (MC) in TCMM [5] framework except that it has additional temporal features which describe TMC temporal existence. When the size of TMCs exceeds a given memory space, similar TMCs have to merge (spatiotemporally) to reduce memory space. In offline phase, any density based approach can be used to cluster temporal micro clusters to generate macro clusters, a response to user request. Note that CC_TRS is used to cluster the trajectories of free moving object.

The main contributions of this paper are as follows:

(i) Propose the concept of temporal micro cluster (TMC) which means TMC exists for a period of time starting from creation time. Clustering task will take into account TMC as long as it exists.

(ii) Define new data structure for TMC.

(iii) Evaluation of proposed algorithm on real data sets shows ability to maximize the cluster quality and minimize the execution time compared with existing algorithms.

The rest of this article is organized as follows. Section 2 presents the data stream clustering algorithms related works. Section 3 presents problem definitions. Section 4 proposed algorithm *CC_TRS*. Section 5 presents performance evaluation. Finally, Section 6 concludes the article.

2. Related Works

Trajectory stream clustering aims to find out representative paths and common tendencies that are shared by group of moving objects. Numerous clustering methods have been presented for static data sets of trajectory; these methods can be classified into five main categories [6, 7]: spatial based clustering [8], time dependent clustering [9], partition and group based clustering [10], uncertain trajectory clustering [11], and semantic trajectory clustering [12].

Many researches have been conducted for stream clustering of data. Aggarwal et al. [13] proposed CluStream framework for clustering evolving data stream; CluStream uses the concept of pyramidal time frame in conjunction with micro clustering approach. However, the CluStream framework does not handle trajectory stream data. Elnekave et al. [14]

presented an incremental clustering algorithm for finding evolving groups of similar mobile objects in spatiotemporal data. In this algorithm, each trajectory is represented by set of minimal bounding box (MBB), the entire overlapping between two trajectories. MMBs represent the similarity between them. The algorithm uses a developed version of incremental K-mean algorithm to cluster moving object trajectories. Jensen et al. [15] presented a disk-based algorithm for continuous clustering of moving object. The algorithm employs clustering features structure that can be updated incrementally. Moving object may be deleted from or inserted into a moving cluster during a period of time. Next, the approach merges and splits the clusters through monitoring their compactness. Li et al. [5] suggested the TCMM framework which consists of two parts: online micro clustering and offline macro clustering for incremental trajectories clustering. Online micro clustering stores statistical information of similar trajectory segments in cluster features (CF) data structure and updates CFs when new batch of segments is added. Similar CFs are merged to solve memory limitation issue. Offline clustering is implemented on the set of micro clusters based on density based clustering when user sends request to see the clustering results. Some studies use optimization strategies such as indexes or pruning to minimize search and enhance the efficiency of clustering. Yu et al. [16, 17] proposed ConTraClu algorithm to cluster continuous high speed trajectories data stream and discover moving pattern such as flock. The algorithm consists of online clustering of trajectory segments depending on density based approach and updating process of closed clusters depends on bi-Tree index. Da Silva et al. [18, 19] proposed an incremental algorithm for trajectory data stream. The algorithm uses a micro group structure to truck moving object and its evolution at consecutive time windows. Micro group describes the relationship among moving objects and evolves (merge or split) in the next time period. Mao et al. [2] produce two-stage framework TSCluWin over sliding window model. During the first stage, sufficient summarized information of the micro clusters is stored and maintained continuously in EF data structure. During the second stage, a small number of micro clusters are produced depending on micro clusters. There also exist some different approaches but they deserve to be mentioned. Costa et al. [20] interpret trajectory as a discrete time signal and use Fourier transform to measure the similarity between two trajectories.

3. Problem Definition

In this section, we will define some notations.

Definition 1 (temporal micro cluster (TMC)). A vector summarized the spatiotemporal features for a set of similar directed segments. TMC is of the following form ($\mathrm{LS}_{\mathrm{cen}}$, $\mathrm{SS}_{\mathrm{cen}}$, $\mathrm{LS}_{\mathrm{th}}$, $\mathrm{SS}_{\mathrm{th}}$, LS_l, SS_l, T, E, and N).

$\mathrm{LS}_{\mathrm{cen}}$: the linear sums of the line segments center points.

$\mathrm{SS}_{\mathrm{cen}}$: the square sums of the line segments center points.

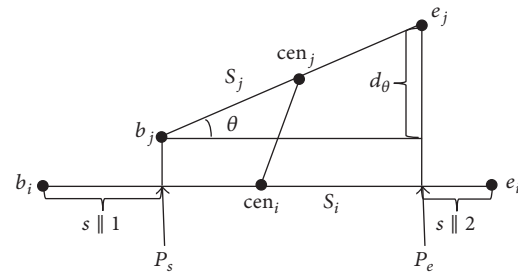


Figure 4: Three components of the distance function.

$\mathrm{LS}_{\mathrm{th}}$: the linear sums of the line segments angles.

$\mathrm{SS}_{\mathrm{th}}$: the square sums of the line segments angles.

LS_l: the linear sums of the line segments lengths.

SS_l: the square sums of the line segments lengths.

T: TMC creation time.

E: True if TMC is expired, otherwise false.

N: number of line segments in temporal micro cluster (TMC).

Note that the lifespan of TMC starts at TMC.T and ends at TMC.T+ Ω, where Ω is a predefined time threshold.

Definition 2 (representative line of TMC). It represents the spatial average of TMC members. The start and end points of any representative line of TMC can be calculated from its features:

$$P_{\mathrm{start}} = \left(x_{\mathrm{cen}} - \frac{\cos(\theta)}{2}l, y_{\mathrm{cen}} - \frac{\sin(\theta)}{2}l\right)$$
$$P_{\mathrm{end}} = \left(x_{\mathrm{cen}} + \frac{\cos(\theta)}{2}l, y_{\mathrm{cen}} + \frac{\sin(\theta)}{2}l\right), \tag{1}$$

where $x_{\mathrm{cen}} = \mathrm{LS}_{\mathrm{cen\_}x}/N$, $y_{\mathrm{cen}} = \mathrm{LS}_{\mathrm{cen\_}y}/N$, $\theta = \mathrm{LS}_{\mathrm{th}}/N$, and $l = \mathrm{LS}_l/N$.

Definition 3. The distance between representative line of TMC (S_i) and trajectory segment (S_j) is equal to the sum of three components: center distance (d_{cen}), angle distance (d_θ), and parallel distance ($d_\parallel$) as illustrated in Figure 4:

$$\mathrm{Dist}\left(S_i, S_j\right) = d_{\mathrm{cen}}\left(S_i, S_j\right) + d_\theta\left(S_i, S_j\right) + d_\parallel\left(S_i, S_j\right)$$
$$d_{\mathrm{cen}}\left(S_i, S_j\right) = \left|\mathrm{cen}_i - \mathrm{cen}_j\right|, \tag{2}$$

where $|\mathrm{cen}_i - \mathrm{cen}_j|$ represents the Euclidian distance between the centers of two segments (S_i, S_j).

$$d_\theta\left(S_i, S_j\right) = \begin{cases} \left\|S_j\right\| * \sin(\theta), & 0 \le \theta < 90 \\ \left\|S_j\right\|, & 90 \le \theta \le 180. \end{cases} \tag{3}$$

Note that $|S_i|$ and $|S_j|$ denote the length of S_i and S_j, respectively. θ represents the smallest angle between S_i and S_j, and the range of θ is within $[0, 180]$.

$$d_\parallel\left(S_i, S_j\right) = \min\left(s\|_1, s\|_2\right), \tag{4}$$

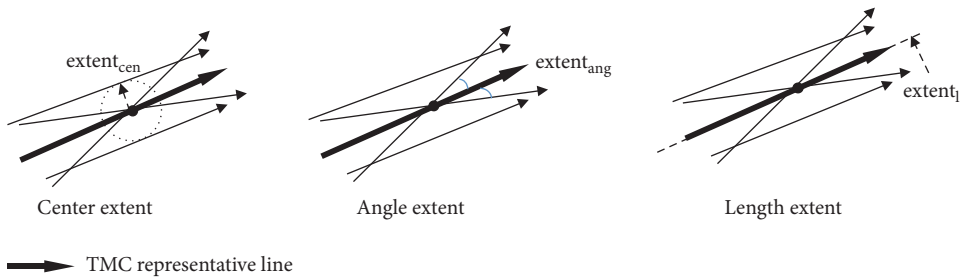


Figure 5: Temporal micro cluster extent.

where $s\|_1$ and $s\|_2$ are the Euclidean distance between the points (b_i, p_s) and (e_i, p_e), respectively. b_i and e_i are the end points of line segments S_i.

Definition 4 (temporal micro cluster extent (δ)). TMC extent is a pointer of its spatial tightness. The extent of TMC comprises three components extent$_{cen}$, extent$_{ang}$, and extent$_l$, since the representation of TMC contains these three parts as illustrated in Figure 5. The extents are the standard deviation which can be computed from its corresponding LS and SS as defined in

$$\delta_\beta = \sqrt{\frac{\left(N * SS_\beta - LS_\beta^2\right)}{N^2}}, \tag{5}$$

where N is the number of line segments in the temporal micro cluster and β represents center, length, and angle.

4. CC_TRS Algorithm

The core idea of CC_TRS algorithm is to specify a lifetime (predefined threshold) for each newly created TMC; the TMCs can only interact with clustering task during their lives. When a new trajectory segment arrives, the CC_TRS algorithm finds the closest nonexpired TMC$_k$ for each segment S_i, since expired TMCs become already far spatially or temporally or both form new coming segments. If the distance between TMC$_k$ and S_i is less than a user distance threshold ($d_{\max}$), S_i will be inserted into TMC$_k$ and update TMC$_k$ information. Otherwise, a new temporal micro cluster TMC$_{new}$ will be created for S_i.

Basically, the algorithm maintains two arrays called TMC and E_TMC to control execution time; the data structure of each array entry is illustrated in Definition 1. CC_TRS algorithm continuously inserts the newly created TMC$_{new}$ into TMC array with its status (existence) and creation time. After a while, the size of TMC array increases and some of TMC$_j$ become expired which affect the efficiency of the algorithm since the most time-consuming part is finding the closest TMC$_i$ in TMC array. Therefore, all expired TMC$_j$ are transferred periodically to E_TMC array to minimize searching time in TMC array. Eventually, if the size of TMC and E_TMC exceeds memory constraint, some TMC will be merged based on their spatiotemporal information. Algorithm 1 illustrates CC_TRS algorithm steps.

Algorithm 1 performs the creation and updating of temporal micro cluster. In lines (6–13) after the arrival of new segment S_j, the algorithm finds the nearest distance between S_j and all nonexpired TMC. TMC$_k$ is nonexpired when the difference between S_j time and TMC$_k$ creation time is less than Ω threshold. In lines (14–19), if the distance (S_j, TMC$_k$) is less than a threshold distance ($d_{\max}$), S_j will be added to TMC$_k$ and update TMC$_k$ information; otherwise a new TMC$_{new}$ will be created for S_j and set TMC$_{new}$.T = S_i.time. Line (21) transfers all expired TMCs to E_TMC array every (Φ) second. In lines (22-23), if the size of both TMC and E_TMC exceeds memory constraint, some TMC will be merged based on spatiotemporal information.

4.1. TMC Merging Algorithm. Spatially, CC_TRS adopts the TCMM [5] technique to merge two TMCs, TCMM suggests taking into consideration the tightness of micro clusters when merging them. Furthermore, TCMM framework gives the priority to merge two loose micro clusters rather than merging tight micro clusters if the distance between their representative lines is equal, since merging very tight micro clusters will break their tightness as in Figure 6(a) while merging two loose micro clusters will not hurt their loss tightness as illustrated in Figure 6(b).

The spatial distance between TMC$_i$ and TMC$_j$ is equivalent to the distance between their representative lines R_i^* with extent δ_i and R_j^* with extent δ_j. Note that extent δ is used to strengthen the similarity of two loose micro clusters in order to give them the priority to merge as illustrated in Figure 7. The distance between R_i^* and R_j^* contains three parts: center distance, angle distance, and parallel distance:

$$\begin{aligned} \operatorname{dist}\left(R_i^*, R_j^*\right) &= d_{\text{cen}}\left(R_i^*, R_j^*\right) + d_\theta\left(R_i^*, R_j^*\right) \\ &\quad + d_\parallel\left(R_i^*, R_j^*\right). \end{aligned} \tag{6}$$

The center distance with extent is

$$d_{\text{cen}}^*\left(R_i^*, R_j^*\right) = \max\left(0, \left|\text{cen}_i - \text{cen}_j\right| - \delta_{\text{cen}}^i - \delta_{\text{cen}}^j\right). \tag{7}$$

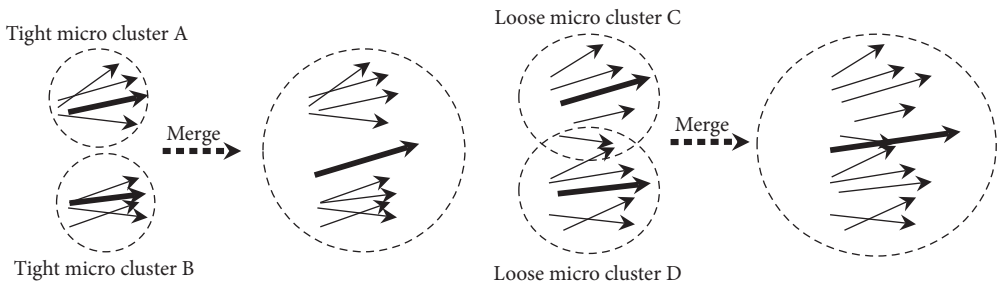
The angle distance with extent is

$$\theta^* = \theta - \left(\delta_\theta^i + \delta_\theta^j\right)$$

$$d_\theta^*\left(R_i^*, R_j^*\right) = \begin{cases} \left\|R_j^*\right\| * \sin\left(\theta^*\right), & 0^\circ \le \theta^* < 90^\circ \\ \left\|R_j^*\right\|, & 90^\circ \le \theta^* \le 180^\circ. \end{cases} \tag{8}$$

(1) Input: Trajectories $T = [\mathrm{Tr}_1, \mathrm{Tr}_2, \ldots, \mathrm{Tr}_m]$
: Temporal Micro clusters TMC = $[\mathrm{TMC}_1, \mathrm{TMC}_2, \ldots, \mathrm{TMC}_n]$
(2) Parameters: $d_{\max}$, Ω, Φ
(3) for every $\mathrm{Tr}_i \in T$ do
(4) for every $S_j \in \mathrm{Tr}_i$ do
(5) Min_dist = inf;
(6) for every TMC_k in TMC
(7) if $\mathrm{TMC}_k.E$ = Expired then next K
(8) if S_i.time- $\mathrm{TMC}_k.T \le \Omega$ threshold % *check the validity of* TMC_k %
(9) dist = distance(TMC_k, S_j);
(10) If dist < Min_dist then Min_dist = distance; index = k
(11) else
(12) TMC_k.Expire = true;
(13) end; {k}
(14) if Min_dist $\le d_{\max}$ then
(15) Add S_j into $\mathrm{TMC}_{\mathrm{index}}$ and update $\mathrm{TMC}_{\mathrm{index}}$ information
(16) else
(17) Create a new micro-cluster $\mathrm{TMC}_{\mathrm{new}}$ for S_j
(18) $\mathrm{TMC}_k.E$ = false; $\mathrm{TMC}_k.T = S_j$.Time;
(19) end;
(20) end; {j}
(21) transfer expired TMC to list E_TMC every Φ second
(22) if size of (TMC, E_TMC) is larger than memory limit then
(23) Merge similar spatio-temporal micro cluster in E_TMC
(24) end; {i}

Algorithm 1: Trajectories micro clustering. CC_TRS algorithm.



(a) Merging tight micro clusters (b) Merging loose micro clusters

Figure 6: Merging micro clusters.

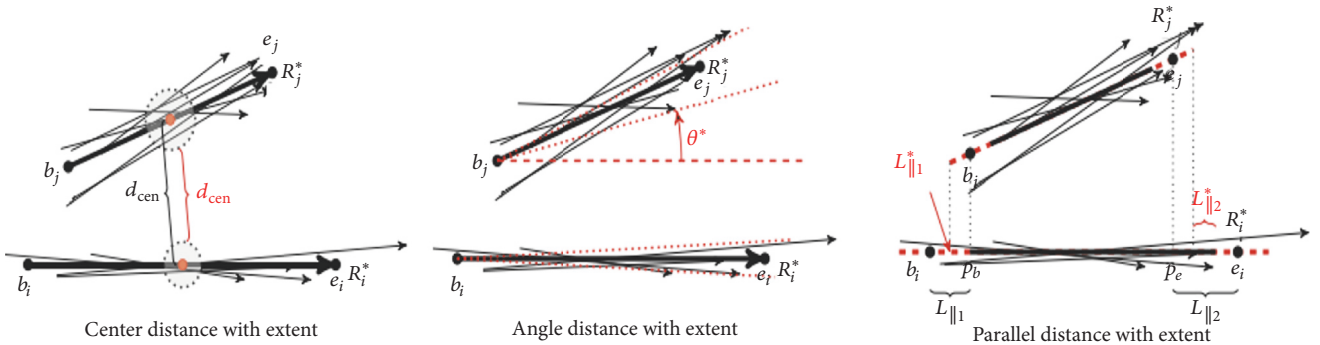
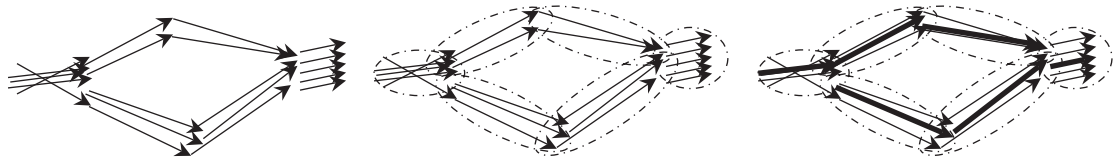
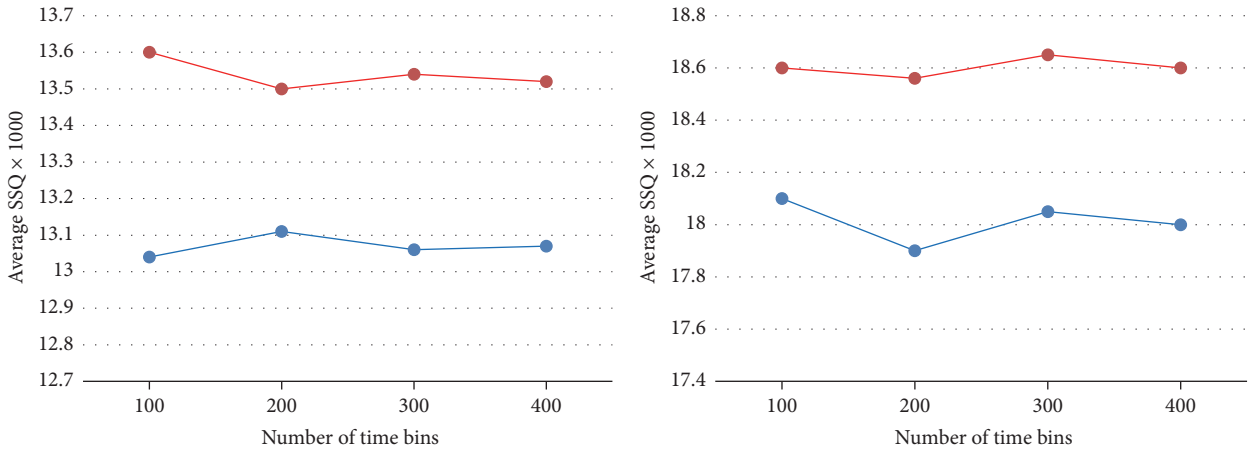


Figure 7: Distance with extent.



(a) Raw trajectory (b) Temporal micro clustering (c) Macro clustering

Figure 8: Macro clustering.



(a) Deer 95 data set (b) ELK 93 data set

Figure 9: Clustering quality comparison CC_TRS versus TCMM.

The parallel distance with extent is

$$d_{\parallel}^{*}\left(R_i^{*}, R_j^{*}\right) = \max \quad \left(0, \min\left(l \parallel 1, l \parallel 2\right) - \left(\delta_{\text{length}}^{i} + \delta_{\text{length}}^{j}\right)\right). \quad (9)$$

When the size of TMCs exceeds the memory limits, the TMCs must be merged to satisfy memory constraint. The merging algorithm of n given TMCs is illustrated in Algorithm 2. Note that CC_TRS maintains temporal and spatial information during merging process while TCMM maintains only spatial features.

4.2. Trajectory Macro Clustering. Macro clustering is evoked when the user requests to see the overall results. Any density based clustering algorithm can be used to achieve macro clustering by replacing the distance between spatial points with the distance between temporal micro clusters as depicted in Figure 8. The distance between temporal micro clusters is defined in (6).

5. Performance Evaluation

In this section, the performance of *CC_TRS* algorithm is tested and compared with TCMM framework and ConTraClu. Two real data sets called Elk1993 and Deer1995 are used; the Elk 93 has 33 trajectories and 47204 points, while Deer 95 has 32 trajectories and 20065 points. Any trajectory segmentation algorithm such as in [10, 21] can be used to divide each trajectory to set of segments. Matlab R2012b and excel 2013 were used to implement the algorithm and plot the charts.

5.1. Clustering Quality Evaluation. The sum of square distance SSQ was used to compare the clustering quality results of CC_TRS with TCMM. The SSQ of n trajectory segments is equal to the sum of square distances between segment S_i and its closest TMC representative line L_i as illustrated in (10) and (11); the value of distance threshold ($d_{\max}$) is set to 600:

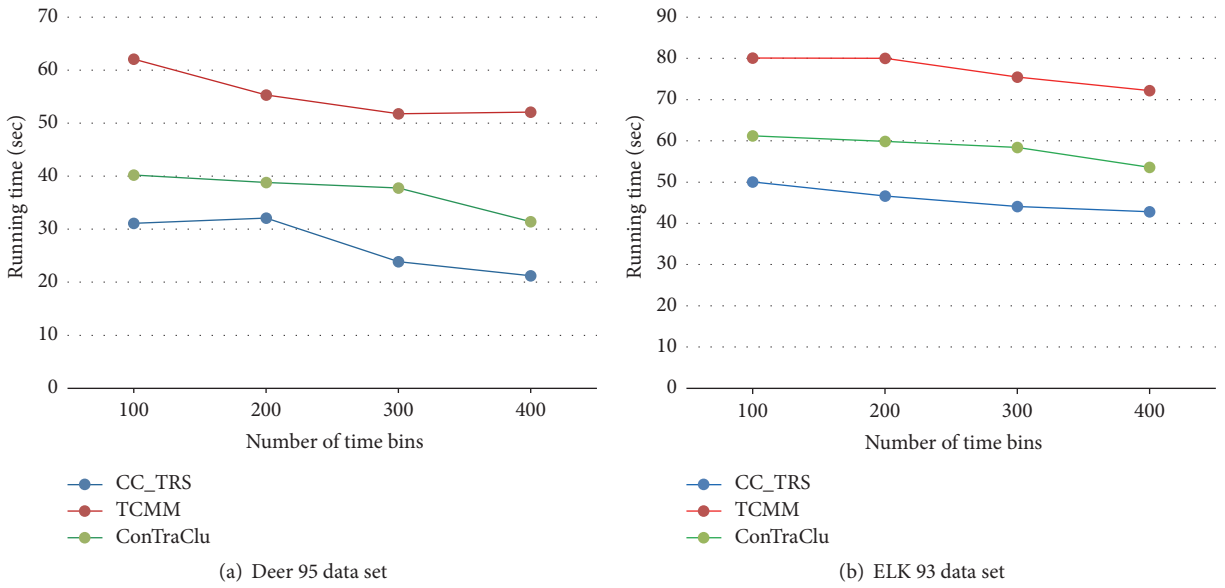
$$\text{SSQ} = \sum_{i=1}^{n} d^2\left(S_i, L_i\right) \quad (10)$$

$$\text{Average SSQ} = \frac{\text{SSQ}}{n}. \quad (11)$$

Figure 9 shows an improvement in clustering quality results of CC_TRS compared with TCMM using data sets Deer 95 and ELK 93. For Deer 95 data set, the maximum improvement is 4.7% when number of time bins is equal to 100, while minimum improvement is 3.75% when number of time bins is equal to 200 as illustrated in Figure 9(a). For ELK 93 data set, the maximum improvement is 3.8% when number of time bins is equal to 200, while minimum enhancement is 3.5% when number of time bins is equal to 100 as depicted in Figure 9(b). Therefore, the improvements of clustering quality range within 3.5–5% (the smaller SSQ, the better clustering quality).

Input: set of temporal micro clusters temporally arranged
TMC = [$TMC_1, TMC_2, \ldots, TMC_n$]
(1) calculate the spatial similarity between every two consecutive (TMC_i, TMC_{i+1})
(2) Arrange the similarity from the most similar to least similar TMCs
(3) Merge most similar TMCs until the size of TMCs satisfy the memory limits.

Algorithm 2: Temporal micro clusters (TMCs) merging.



(a) Deer 95 data set (b) ELK 93 data set

Figure 10: Efficiency comparison (running time).

5.2. Efficiency Evaluation (Time and Memory Space). The CC_TRS was compared with TCMM and ConTraClu algorithms to evaluate its efficiency in terms of execution time and space requirements. TCMM and ConTraClu divide stream data to set of time bins and cluster each time bin separately, while CC_TRS allocates lifespan for each new created cluster. To make the comparison fair, we set cluster life equal to time bin as in Figures 1 and 3. Equation (12) can be used to calculate cluster life (time bin):

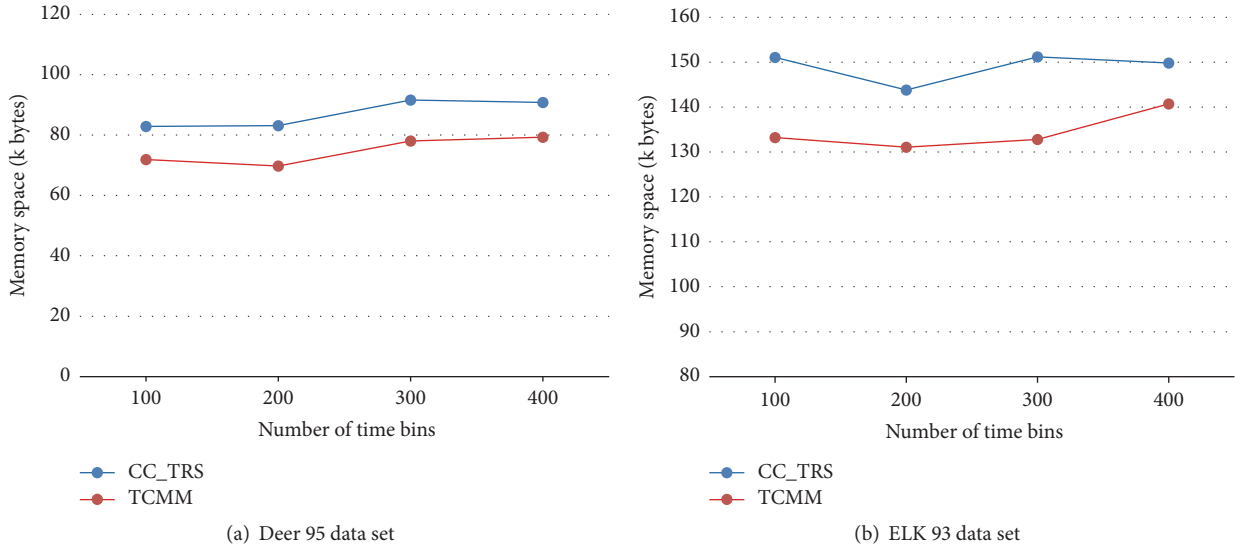
$$\mathrm{CL} = \frac{\left(T_f - T_l\right)}{\mathrm{NOB}}, \tag{12}$$

where T_f and T_l are the stamp time of the first and last segments in data stream and NOB is the number of time bins which are specified by user threshold. The stamp times of first and last segments for ELK 93 data set are (4.6896×10^4) and (4.9320×10^4) minutes and for Deer 95 data set are (6.3553×10^4) and (6.6840×10^4) minutes. The algorithms are run several times for different values of NOB (100, 200, 300, and 400); Figures 10(a) and 10(b) illustrate that the execution time of the CC_TRS is less than TCMM and ConTraClu.

On the other hand, the tests show that *CC_TRS* needs 10–15% higher of memory space than TCMM as illustrated in Figures 11(a) and 11(b). The size of micro cluster in TCMM is 28 bytes, since the micro cluster has 7 fields and each field declared using uint32 Matlab variable (4 bytes). While TMP size is 32 bytes and one bit since it has two extra fields, the first field T is used to save creation time of TMC and a logical variable E is used to save the status of TMC (expired, nonexpired). It is obvious that TMC size is equal to 1.15 of micro cluster size; therefore, most of the extra memory required by CC_TRS comes from the additional temporal field in TMC data structure. Note that we compare the memory requirements of CC_TRS with only TCMM since both algorithms have similar data structure.

5.3. Parameter Φ Effect. To explain the effect of Φ parameter on execution time of CC_TRS, the algorithm is run several times with different values of Φ range within 1000–3500 seconds. We set the value of NOB (400) and $d_{\max}$ (600) since these values give minimum execution time for Deer 95 data set as shown in Figure 10(a). Figure 12 shows that minimum execution time is achieved when Φ is equal to 2000 seconds.

5.4. Parameter d_{max} Effect. In this section, we describe the effects of $d_{\max}$ on clustering quality and running time depending on Deer 95 data set. The smaller the $d_{\max}$, the better the clustering quality but it requires longer execution time. On the other hand, the larger $d_{\max}$, the faster running time but more information will be lost in micro clustering. For example, the clustering quality when $d_{\max} = 400$ (red line) is better than its value when $d_{\max} = 700$ (blue line) as illustrated in Figure 13(a), while the running time is higher for



(a) Deer 95 data set (b) ELK 93 data set

Figure 11: Efficiency comparison (memory).

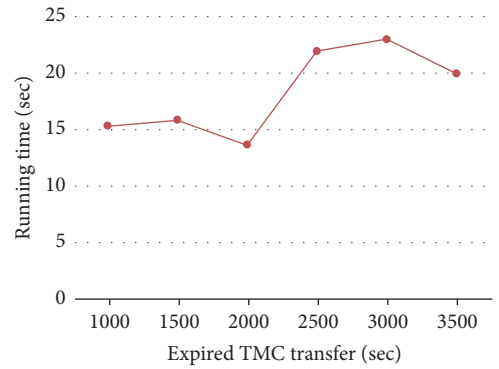
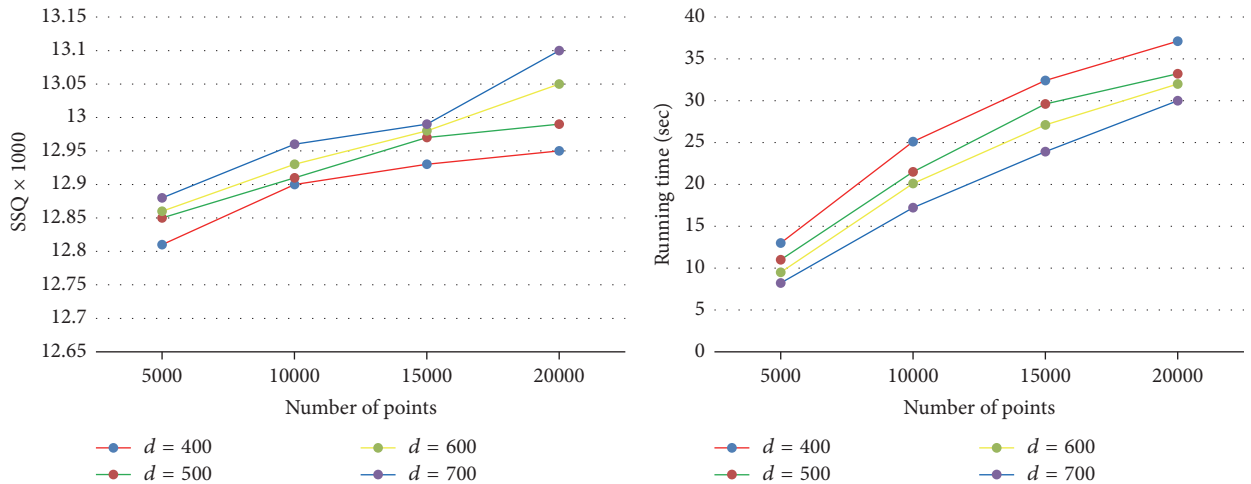


Figure 12: The effect of Φ on execution time.



(a) Effect of $d_{\max}$ on SSQ (b) Effect of $d_{\max}$ on running time

Figure 13: The sensitivity of parameter $d_{\max}$ (SSQ versus time).

$d_{\max} = 400$ as illustrated in Figure 13(b). As a consequence, a trade-off between clustering quality and running time is required to get the best results.

6. Conclusion

In this article, CC_TRS algorithm is proposed for the clustering of trajectories stream data for free moving object. The algorithm consists of two phases: online phase and offline phase. In online phase, CC_TRS algorithm suggests a lifespan technique which assigns a lifetime for each newly created temporal micro cluster instead of time window techniques which divide stream data to time bin and cluster each time window separately. In offline phase, density base clustering approach is used to demonstrate clustering results on user demand. The tests of CC_TRS on two data sets Deer 95 and ELK 93 minimize running time (50% and 20%) compared with TCMM and ConTraClu, respectively. Besides that, the clustering quality is improved by 3.5–5% compared with TCMM based on the sum of square distance SSQ. On the other hand, CC_TRS algorithm needs higher memory space by 10–15% compared to TCMM framework since the data structure of temporal micro cluster has extra temporal fields.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, article 29, 2015.

[2] J. Mao, Q. Song, C. Jin, Z. Zhang, and A. Zhou, *TSCluWin: Trajectory Stream Clustering over Sliding Window*, vol. 9643 of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[3] Z. Galić, *Spatio-temporal data streams*, Springer Briefs in Computer Science, Springer, New York, NY, USA, 2016.

[4] C. C. Aggarwal and C. K. Reddy, *Data Classification: Algorithms and Applications*, CRC Press, New York, NY, USA, 2014.

[5] Z. Li, J.-G. Lee, X. Li, and J. Han, "Incremental clustering for trajectories," in *Database Systems for Advanced Applications*, vol. 5982 of *Lecture Notes in Computer Science*, pp. 32–46, Springer, Berlin, Germany, 2010.

[6] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2016.

[7] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo, "Spatio-temporal clustering," in *Data Mining and Knowledge Discovery Handbook*, pp. 855–874, Springer, Boston, MA, USA, 2010.

[8] C.-C. Hung, W.-C. Peng, and W.-C. Lee, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *VLDB Journal*, vol. 24, no. 2, pp. 169–192, 2015.

[9] M. D. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *Journal of Intelligent Information Systems*, vol. 27, no. 3, pp. 267–289, 2006.

[10] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*, pp. 593–604, ACM, Beijing, China, June 2007.

[11] J. Chen, Q. Huo, P. Chen, and X. Xu, "Sketch-based uncertain trajectories clustering," in *Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '12)*, pp. 747–751, May 2012.

[12] X. Wang, G. Li, G. Jiang, and Z. Shi, "Semantic trajectory-based event detection and event pattern mining," *Knowledge and Information Systems*, vol. 37, no. 2, pp. 305–329, 2013.

[13] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases (VLDB Endowment '03)*, vol. 29, Elsevier, 2003.

[14] S. Elnekave, M. Last, and O. Maimon, "Incremental clustering of mobile objects," in *Proceedings of the Workshops in Conjunction with the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 585–592, April 2007.

[15] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1161–1173, 2007.

[16] Y. Yu, Q. Wang, and X. Wang, "Continuous clustering trajectory stream of moving objects," *China Communications*, vol. 10, no. 9, Article ID 6623510, pp. 120–129, 2013.

[17] Y. Yanwei, Q. Wang, X. Wang, H. Wang, and J. He, "Online clustering for trajectory data stream of moving objects," *Computer Science and Information Systems*, vol. 10, no. 3, pp. 1293–1317, 2013.

[18] T. L. C. Da Silva, K. Zeitouni, and J. A. F. De Macedo, "Online clustering of trajectory data stream," in *Proceedings of the 17th IEEE International Conference on Mobile Data Management (IEEE MDM '16)*, pp. 112–121, June 2016.

[19] T. L. C. da Silva, T. L. Coelho, K. Zeitouni, J. A. F. de Macêdo, and M. A. Casanova, "CUTiS: optimized online ClUstering of Trajectory data Stream," in *Proceedings of the 20th International Database Engineering & Applications Symposium*, pp. 296–301, ACM, 2016.

[20] G. Costa, G. Manco, and E. Masciari, "Dealing with trajectory streams by clustering and mathematical transforms," *Journal of Intelligent Information Systems*, vol. 42, no. 1, pp. 155–177, 2014.

[21] M. Riyadh, N. Mustapha, N. Sulaiman, and N. B. Sharef, "ONF-TRS: On-line Noise Filtering Algorithm for Trajectory Segmentation Based on MDL Threshold," *Journal of Artificial Intelligence*, vol. 10, no. 1, pp. 42–48, 2016.