CUTiS*: optimized online ClUstering of Trajectory data Stream

Ticiana L. Coelho da Silva
Federal University of Ceará, Brazil
ticianalc@ufc.br

Karine Zeitouni
Université de Versailles-St-Quentin, France
karine.zeitouni@uvsq.fr

José A. F. de Macêdo
Federal University of Ceará, Brazil
jose.macedo@lia.ufc.br

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

ABSTRACT

Recent approaches for online clustering of moving objects location are restricted to instantaneous positions. Subsequently, they fail to capture the behavior of moving objects over time. By continuously tracking sub-trajectories of moving object at each time window, it becomes possible to gain insight on the current behavior and potentially detect mobility patterns in real time. In our previous work [1], we proposed CUTiS, an incremental algorithm for discovering and maintaining the density-based clusters in trajectory data streams, while tracking the evolution of the clusters. This paper extends [1] to CUTiS* by proposing an indexing structure for sub-trajectory data based on a space-filling curve. The proposed index improves the performance of our approach without losing quality in the clusters results as we show in our experiments conducted on a real dataset.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous; H.3 [**Information Storage and Retrieval**]: Miscellaneous

Keywords

Clustering, Sub-trajectory & Data Stream

1. INTRODUCTION

Nowadays, many services involve tracking moving objects (e.g., persons, vehicles, animals) to report their trajectory continuously (e.g., every second or every minute). Analyzing these data while they are generated may bring a real added-value in the comprehension of the city dynamics and the detection of regularities as well as anomaly, which is essential for decision making. Among these patterns, we consider in this paper the (sub)trajectory clustering and its evolution. Such discovery may help the search for effective re-engineering of traffic, or dynamically detecting events or incidents at a city level.

There exist approaches for online clustering of moving objects position, but they are restricted to instantaneous positions. Subsequently, they fail to capture the displacement behavior along time. The continuous analysis of the sub-trajectory data while it arrives can provide much better comprehension of current cluster patterns and their evolution between consecutive periods of time. It becomes possible to gain insight on the current behavior and potentially detect suspicious behaviors in real time. Finding clusters over these data stream in (quasi) real time is quite challenging because all tracked moving objects may change their positions every time, clusters may also change accordingly and new moving objects may appear as well as others may stop and disappear. These changes may affect clusters formation.

CUTiS (standing for ClUstering Trajectory Stream), presented in [1], is an incremental sub-trajectory clustering algorithm based on time, space and direction distance function for trajectory data stream. Moreover, it proposes a new structure, called micro-group, to represent the relationship among moving objects and to track the evolution of clusters/groups, e.g., merge or split in the next time period. In this paper, we propose an indexing structure for sub-trajectory data based on a space-filling curve. This method has the property of mapping a multidimensional space to one-dimensional space such that, for two objects that are close in the original space, there is a high probability that they will be close in the mapped target space. We take advantage of this property to optimize [1] and develop CUTiS*. We also conducted new experiments comparing it to [1] with a real dataset to evaluate its effectiveness and efficiency.

This paper is organized as follows. Section 2 provides a running example to explain CUTiS [1]. Section 3 discusses the indexing structure for trajectory data proposed in CUTiS*. Section 4 shows our experimental evaluation. Section 5 summarizes related work, before the conclusion.

2. CUTIS

CUTiS follows four steps (firstly proposed in [2] and extensively discuss in [1], [3]): (i) application of a distance

IDEAS '16, July 11-13, 2016, Montreal, QC, Canada

DOI: http://dx.doi.org/10.1145/2938503.2938516

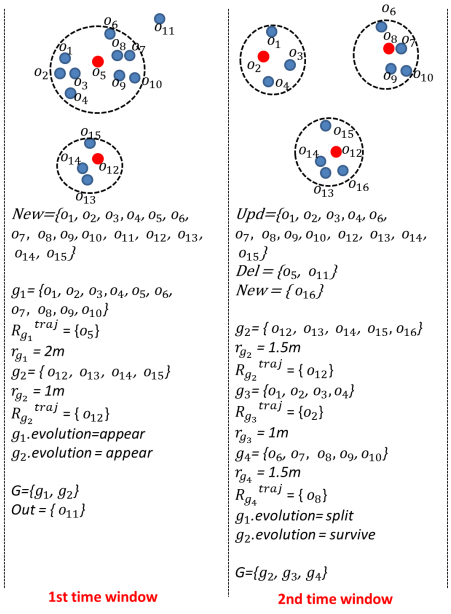


Figure 1: Micro-Group Maintenance

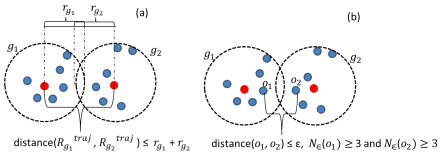


Figure 2: Micro-Group Sub-trajectory Clustering

function; (ii) choice of representative trajectories; (iii) maintenance of the micro-groups; and (iv) discovery of sub-trajectory clusters. In this paper, we have chosen to explain it by a running example.

Table 1 lists the notation used throughout this paper. Figure 1 shows the running process of micro-group maintenance algorithm. To create a micro-group, a moving object is randomly picked and checked if its sub-trajectory can be a representative of a new micro-group. Then, the micro-group is derived as the objects that vote for the chosen representative. Suppose the $\epsilon = 1$ meter, $\rho = 0.17$ (this means the maximum value for the micro-group radius is around to 2 meters) and $\tau = 3$. At the first time window, two micro-groups are created, then both micro-groups evolution are appear. During the second time window, the moving objects o_5 and o_{11} stop to send their positions, so they should be deleted. The deletion of o_5 triggers to split g_1 into two micro-groups g_3 and g_4. This process is done by choosing a representative trajectory from the moving objects that belonged to g_1, and the moving objects in g_1 that vote for the chosen representative. It ends when it is not possible to create another micro-group with g_1 data. The micro-group g_2 survives in the second time window, furthermore a new moving object o_{16} starts to send its positions and it is added to the micro-group g_2. This means the moving object o_{16} can be represented by the representative trajectory R_{g2}^{traj} of the micro-group g_2.

Notation	Explanation
ϵ	the distance threshold
τ	the size or density threshold
ρ	representativeness threshold
New	the set of new moving objects
Del	the set of moving objects deleted
Upd	the set of moving objects updated
o_i	a moving object
Out	the set of moving objects outlier
g_i	a micro-group
r_{g_1}	micro-group radius (the distance value between the representative trajectory of g_i and the farthest moving object sub-trajectory in g_i)
$R_{g_i}^{traj}$	the pair $\{o_j, ST_{j,i}\}$ which has the representative trajectory of micro-group g_i
G	the set of micro-group

Table 1: Table of Notations

Each micro-group is density-based. Hence, it is suitable to find sub-trajectory density-based clusters by merging two or more micro-groups into one cluster. Our clustering algorithm is a variant of DBSCAN (based on two parameters ϵ and τ) and has two phases: (i) Find the Merge Candidates and (ii) Density-based Clustering. In the first phase, two micro-groups are candidates to be merged into a density-based cluster, if they intersect or are tangent to each other w.r.t. their circular area. By following Figure 2(a), g_1 and g_2 are micro-groups and r_{g_1} and r_{g_2} are their respectively radius. If the $distance(traj[R_{g1}^{traj}], traj[R_{g2}^{traj}]) \leq r_{g1} + r_{g2}$ is satisfied, then g_1 and g_2 are candidates to be merged. In the second phase, our approach finally finds the clusters using the merge candidates. If g_1 is a candidate to merge with g_2, the algorithm checks if they are direct density reachable (as show in Figure 2(b), there are $o_1 \in g_1, o_2 \in g_2$ such that $distance(o_1, o_2) \leq \epsilon$, $|N_\epsilon(o_1)| \geq \tau$ and $|N_\epsilon(o_2)| \geq \tau$), so the algorithm can merge g_1 and g_2 into the same cluster.

By pruning the merge candidates, our approach saves computation and the final clustering result is approximately that which is produced by applying the original DBSCAN w.r.t. ϵ and τ as presented in the experiments section. However, our focus is to decrease the computational cost in this paper, mainly in two costly steps: (i) Checking if a moving object sub-trajectory can be a representative of a micro-group, since it is necessary to compare it with the whole dataset; (ii) Retrieving the neighbors from a moving object sub-trajectory. Section 3 presents an indexing structure that helps to optimize these steps.

3. CUTIS*: INDEXING TRAJECTORY DATA

3.1 Overview

To create a new micro-group, our initial approach randomly picks an unvisited moving object o_j and checks if its sub-trajectory can be a representative trajectory. This process corresponds to first executing a range query in order to retrieve the neighbors of o_j ($N_\epsilon(o_j)$) and then checking the density of the neighborhood. This process can be costly, since the sub-trajectory o_j is compared with the whole dataset. Moreover, in the worst case, o_j might not satisfy the condition to be a representative trajectory of a new micro-group. We aim at optimizing the search for a representative trajectory and also the range query execution by filtering out

some moving objects sub-trajectories. To achieve this goal, we employ an indexing technique for moving object trajectory data. First, CUTiS* maps the trajectory data into a spatial grid SG_i which was proposed in [4]. This mapping allows efficient filtering of the trajectories in a given cell $C_{m,n}$ of SG_i, where m and n indicate the position of the cell in the grid space. Then, we use a space-filling curve to index the grid cells.

A space-filling curve has the property of mapping a multidimensional space to an one-dimensional space such that, for two objects that are close in the original space, there is a high probability that they will be close in the mapped target space. We take advantage of this property to optimize range queries from a moving object sub-trajectory. Our approach uses Hilbert curve [5], but other types of space-filling curves can be used as well to index the spatial cells (e.g., z-order). In this way, each grid cell $C_{m,n}$ is covered with the Hilbert curve corresponding to the grid granularity and labeled with the obtained Hilbert index. At last, CUTiS* stores the mapping between moving object sub-trajectories and the corresponding Hilbert indexes, and maintains a B-tree index on this data using the attribute combination (Hilbert index, timestamp). Our experiments show how fast it is to create the proposed index. In the following, we present an optimized algorithm for identifying a representative trajectory, which processing is based on the proposed index.

3.2 Choice of a Representative Candidate

The candidate set is derived from all the moving objects o_j which have a mapping with a dense Hilbert Index (or a dense grid cell). We consider a dense Hilbert Index (or a dense grid cell) when it is associated to at least τ different moving objects. To take advantage of the spatial proximity that the space-filling curve techniques tries to preserve, the candidate set is also formed by the moving objects which have some position in the adjacent cells of a dense grid cell. Two cells $C_{m,n}$ and $C_{k,l}$ are adjacent if and only if $|m-k| = 1$ or $|n-l| = 1$. In this way, CUTiS* picks from this candidate set a moving object sub-trajectory (instead of from the whole dataset) to derive a new micro-group.

3.3 Range Query

In [1], our distance function considers time, space and direction. Each time window $i = [t, t+\delta t]$ is divided into n timestamps, so each moving object sub-trajectory has a position update for each timestamp. In order to optimize the range query from a moving object o_j sub-trajectory, CUTiS* queries the B-tree index (based on the tuple {Hilbert Index, timestamp}) and retrieves the moving objects which are "possible neighbors" of o_j.

Indeed, not all "possible neighbors" satisfy the distance criteria, due to the slight difference between the range filtering in the index and the exact distance between sub-trajectories. Whereas our distance function considers the entire sub-trajectory geometry, the range query is only based on the instantaneous spatial positions. We also consider the direction in the distance, but not in the index structure. In spite of its coarse filtering capability, the index provides a significant optimization of our approach. The range query from o_j is based on the following idea:

1. Each position $p = (x, y, t)$ of o_j sub-trajectory has an associated timestamp t and a Hilbert Index h_i;
2. CUTiS* queries a B-tree index to find the moving objects which have at least one position associated with the same timestamp t and Hilbert Index h_i;
3. As the neighbors of o_j might be in adjacent grid cells, CUTiS* also queries the moving objects associated with the Hilbert Index of the adjacent cells (for the same timestamps associated with o_j positions);
4. Since each moving object o_k retrieved from the index is a "possible neighbor", CUTiS* checks if $distance(ST_{j,i}, ST_{k,i}) \leq \epsilon$ is satisfied, then o_k is added to $(N_\epsilon(o_j))$.

The same range query procedure detailed above is applied in almost all CUTiS* steps: in the choice of representative trajectories, in the maintenance of the micro-groups and in the discovery of sub-trajectory clusters. The next section shows the gain of our approach by using the index.

4. EXPERIMENTS

To the best of our knowledge, there is no other incremental trajectory clustering algorithm for trajectory data streams which tracks the moving object trajectory and clusters evolution as time goes. However, for validation purposes, we employ as a baseline some state-of-the-art methods, TraClus [6] and DBSCAN, with the following changes: (i) at each time window, we run DBSCAN and TraClus giving as input only the sub-trajectories of the current time window and not the complete trajectory; (ii) we also adapted TraClus and DBSCAN to use our distance function to take into account the distance in time, space and direction. The clustering parameters $eps = \epsilon$ and $minPoints = \tau$ are the same for CUTiS*, DBSCAN and TraClus implementation. Our proposed index to accelerate the range queries was also used in the DBSCAN implementation.

4.1 Evaluation Metrics

The DBSCAN results are used as the ground truth (since it is the most representative density-based clustering and both TraClus and CUTiS* share many characteristics with it) to test the effectiveness (recall and precision) of CUTiS* for the clusters found. We matched each cluster C_i^{mg} found by CUTiS* with only one DBSCAN cluster C_i^{dbscan}. On the matching process, we applied Jaccard Similarity to compare the clusters based on their moving objects labeled as core (since the set of core moving objects is a deterministic result in DBSCAN). In this way, the cluster C_i^{mg} matches with one DBSCAN cluster that presents the higher Jaccard similarity value. The recall and precision are applied for each matched pair of clusters to measure the quality of clusters result.

The recall for $\{C_i^{mg}, C_i^{dbscan}\}$ is the proportion of correctly core moving objects classified by CUTiS* in C_i^{mg} (true positives) over the core moving objects classified by DBSCAN in C_i^{dbscan} (true positives+false negatives). The precision measure for $\{C_i^{mg}, C_i^{dbscan}\}$ is the proportion of moving objects correctly classified as core moving objects by CUTiS* in C_i^{mg} (true positives) over all the core moving objects classified in C_i^{mg} by CUTiS* (true positives+false positives). If two or more clusters of CUTiS* match with the same DBSCAN cluster, we gather them in the same matching. In this paper, we report the average recall and precision for all the matched clusters. Let $Core_i^{dbscan}$ and $Core_i^{mg}$ be the set of core moving objects in C_i^{dbscan} and C_i^{mg} clusters, respectively.

Table 2: table

	Number of Clusters				
Time Window	1	2	3	4	5
Scenario I	14	10	12	15	14
Scenario II	14	15	22	16	15

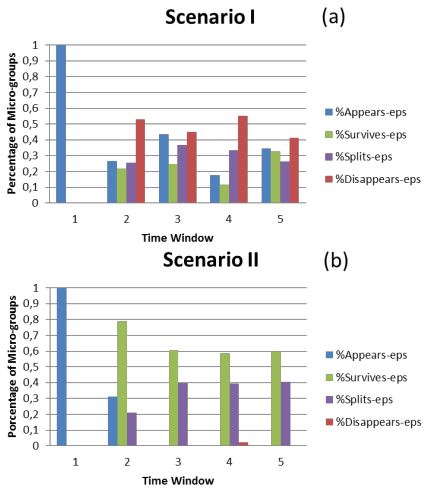


Figure 3: Micro-group evolution pattern found by CUTiS* during 5 tracked time windows.

1. $Recall_{outlier} = \frac{|Out_i^{dbscan} \cap Out_i^{mg}|}{|Out_i^{dbscan}|}$

2. $Precision_{outlier} = \frac{|Out_i^{dbscan} \cap Out_i^{mg}|}{|Out_i^{mg}|}$

We also measured the efficiency of CUTiS* against our competitors.

4.2 Experimental Results

The experiments use a subset of a real world GPS recorded data from taxis of Beijing, containing around 4,000 trajectories. Two possible scenarios were investigated. When a moving object stops sending its positions: (i) **Scenario I**: it should be deleted in the next time window or (ii) **Scenario II**: its sub-trajectory positions should be predicted until a timeout is reached (here the timeout is equals to 2 time windows, i.e., 14 minutes). Both scenarios use a time window size of 7 (seven) minutes and the clusters are tracked for 5 (five) sequential time windows. The number of clusters on the 5 tracked time windows is shown in Table 2.

Micro-group Evolution Pattern. CUTiS* captures the micro-group evolution. For both scenarios, Figure 3 reports the percentage of micro-groups that evolve according to each pattern for all the tracking time windows. Each scenario has its peculiarities which affect the micro-groups maintained. In Scenario I, there exist moving objects deleted at each time window; hence the proportion of micro-groups that disappear or split is greater than in Scenario II. However, even if the majority of micro-groups survive and there exist moving objects whose positions are predicted, Scenario II is not far from the real-world objects. Around 30% to 40% of micro-group split in Scenario II.

Effectiveness Analysis. The effectiveness analysis is measured by applying the precision and recall metrics and comparing the clusters in DBSCAN (the ground truth), DBSCAN with index, TraClus and CUTiS*. In our experiments, CUTiS* found the same clusters as DBSCAN when the micro-group radius=*eps*. Figure 4.2 compares the shapes of the clusters in DBSCAN, TraClus and CUTiS*. Figure 4.2 concentrates in only two time windows and each color represents one DBSCAN cluster. We obtain the best matches with DBSCAN, in both time windows, especially with radius=*eps*. The clusters found by DBSCAN with index are omitted in Figure 4.2 to save space.

When micro-groups have radius=3*eps*, the differences are greater. In the second time window, CUTiS* included the clusters represented by black and light blue colors into the same cluster (represented by black color). The clusters represented by red and dark blue colors are also included into the same cluster (represented by dark blue color). This means that a cluster found by CUTiS* contains core objects from two different DBSCAN clusters. Indeed, these core objects can be represented by their micro-group representative trajectory, however the core objects are not density reachable objects (according to DBSCAN definition) to belong to the same cluster.

As expected, CUTiS* presents higher precision and recall for cluster detection when the micro-group radius=*eps* (Figure 5 and 6), since the clusters produced by CUTiS*, in these cases, have similar shapes to DBSCAN clusters. Unlike for TraClus and when the micro-group has radius=3*eps*, both have in all tracked time windows lower precision and recall values. TraClus is based on DBSCAN algorithm [6]. However, TraClus did not produce similar clusters to DBSCAN for all the tracked time windows. Furthermore, there are many moving objects in DBSCAN clusters that were not clustered by TraClus. We can clearly see that some clusters were not found in the second time window (e.g., black, light blue, dark blue colors) in Figure 4.2. The main difference between TraClus and DBSCAN results is the preprocessing phase in TraClus which partitions the trajectories, then it modifies the similarity since it is applied per partition instead of the whole sub-trajectories. So even the prepocessing phase in TraClus increases its performance, the effectiveness of CUTiS* outperforms it.

Even though our proposed index accelerates the performance for DBSCAN (as we will see), some "core" moving objects were not labelled as core objects. Then, using the index in DBSCAN implementation can not guarantee the same result as the original DBSCAN. This fact happens because the index could not retrieve all moving object neighbors in the range query execution. Even if the index retrieves the moving objects which have sub-trajectory positions close to each other, it can not guarantee they will be close in the whole sub-trajectory shape. In this way, by using our proposed index in DBSCAN implementation, it loses accuracy compared to the original DBSCAN.

Efficiency Analysis. Figure 7(a) and (b) compare the running time of DBSCAN, our approach with micro-group radius=*eps* without the index for trajectory data (called "CUTiS-eps-without index") and our approach for micro-group radius=*eps* with our proposed index (called "CUTiS*-eps-with Index"). Our goal here is to show the improvement of CUTiS* due to the proposed indexing scheme. As expected, the performance of our approach using the index is much better than without index. By using the index, the speed-up of CUTIS* compared to CUTIS (without index)

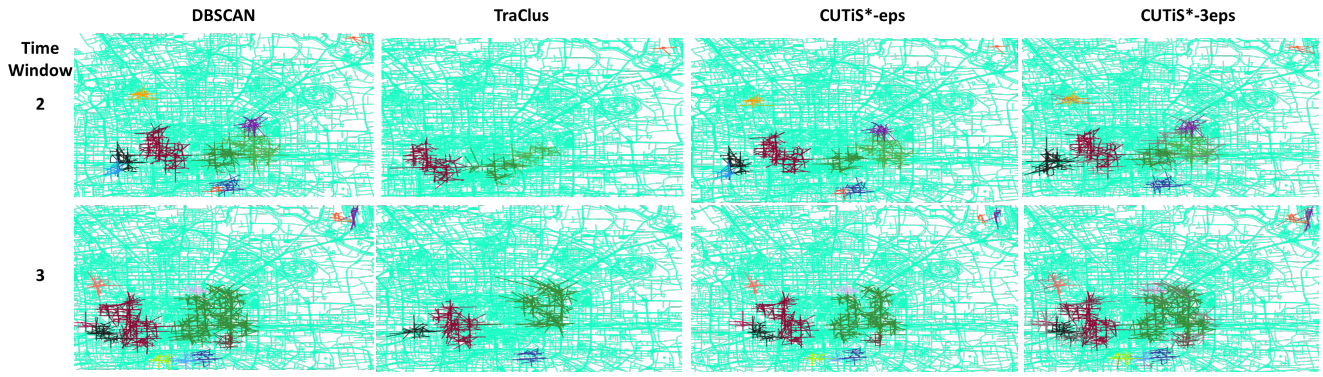
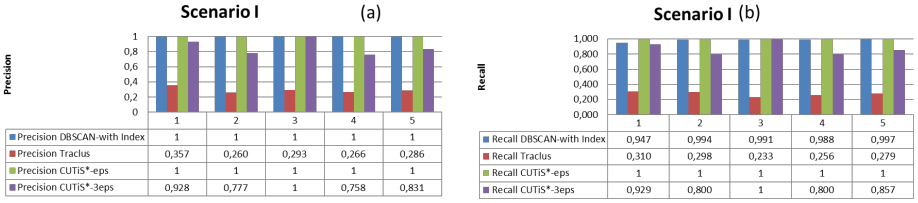


Figure 4: Comparison between clusters shapes of DBSCAN, TraClus and CUTiS* (varying the micro-group radius) for 2 time windows using data of Scenario I.



	1	2	3	4	5
Precision DBSCAN-with Index	1	1	1	1	1
Precision Traclus	0,357	0,260	0,293	0,266	0,286
Precision CUTiS*-eps	1	1	1	1	1
Precision CUTiS*-3eps	0,928	0,777	1	0,758	0,831

	1	2	3	4	5
Recall DBSCAN-with Index	0,947	0,994	0,991	0,988	0,997
Recall Traclus	0,310	0,298	0,233	0,256	0,279
Recall CUTiS*-eps	1	1	1	1	1
Recall CUTiS*-3eps	0,929	0,800	1	0,800	0,857

Figure 5: Effectiveness Analysis Scenario I: (a) Precision and (b) Recall for detecting clusters

ranges from 2.5 to 5. In general, CUTiS* spent 10% of the running time to create the index at each time window (from 30 to 40 seconds). CUTiS* outperforms DBSCAN, since DBSCAN requires to check the density connectivity for each moving object sub-trajectory with the whole dataset, which is computationally costly.

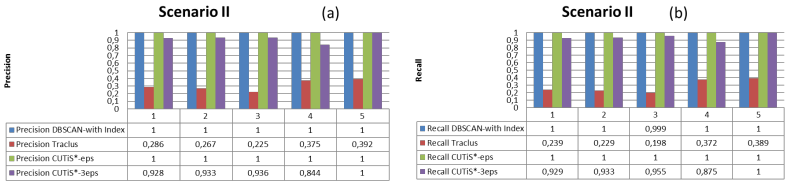
To faithfully compare, we adapted the DBSCAN implementation to use the same index as CUTiS*. Figure 7(c) and (d) show the running time for DBSCAN with index, TraClus and CUTiS* with micro-group radius=*eps* and micro-group radius=3*eps*. As expected, the index also improved the DBSCAN performance. Even though the running time difference between CUTiS* and "DBSCAN-with index" is not large, DBSCAN can not track the evolution of clusters/groups as CUTiS* does and the clusters found by "DBSCAN-with index" are not the same as those of the original DBSCAN (as we discussed before, "DBSCAN-with index" lost accuracy compared to the original DBSCAN). Our results present more efficiency gains than "DBSCAN-with index" when the micro-group radius increases (it maintains less representative trajectories, i.e., a smaller number of micro-groups). TraClus approximated in the experiments most of the time the sub-trajectory to few segments, as we mentioned before. In this case, the distance computation is less costly than considering each sub-trajectory with many segments, as CUTiS* did. The segmentation reduces the computational complexity and this is the main reason why TraClus is much faster to create the clusters.

In general, CUTiS* presented a trade-off between quality and performance, which is influenced by the micro-group radius value. Generally speaking, when the radius has a low value, i.e., the micro-groups size decreases, it leads to maintaining too many representatives (low performance), but our clustering algorithm produces similar result to DBSCAN (high quality). By increasing the micro-group radius, CUTiS* presents high performance but it may lead to misclassify outliers (reducing the quality). Moreover, by using our proposed index, CUTiS* improved its performance when compared with CUTiS (i.e., without index).

5. RELATED WORK

Trajectory Clustering. There are some works related to clustering trajectory data [7, 8]. However the major problem is that those approaches tend to generate clusters for the entire trajectory dataset, instead of the most recent time window. Hence, the fine-grained spatio-temporal relationships between moving objects are lost. Papers [6, 9, 10] cluster sub-trajectory data, but [6] focused on spatial criteria and ignored the time dimension, whereas [9, 10] only considered road network constrained movement. Also they are not suitable for incremental data since clusters are re-calculated from scratch every time. The work [11] proposed efficient algorithms for maintaining and updating the clusters when new trajectories are received. However, it does not consider the temporal aspects of the trajectories. As such, moving objects whose trajectories are in the same cluster may not actually stay together temporally. Paper [12] proposed a trajectory clustering algorithm to extract some patterns similar to convoy pattern. The paper [13] proposes an online trajectory clustering over sliding window composed by two components: a micro-clustering that extracts the summary of trajectory stream in the window, and a macro-clustering component that re-clusters the previously extracted summaries according to user's request. However both papers do not track the cluster/group evolution neither the moving objects trajectories update in time.

Moving Objects Clustering. The approaches proposed in [14, 15, 16] cluster moving objects based on the object



	1	2	3	4	5
Precision DBSCAN-with Index	1	1	1	1	1
Precision Traclus	0,286	0,267	0,225	0,375	0,392
Precision CUTiS*-eps	1	1	1	1	1
Precision CUTiS*-3eps	0,928	0,933	0,936	0,844	1

	1	2	3	4	5
Recall DBSCAN-with Index	1	1	0,999	1	1
Recall Traclus	0,239	0,229	0,198	0,372	0,389
Recall CUTiS*-eps	1	1	1	1	1
Recall CUTiS*-3eps	0,929	0,933	0,955	0,875	1

Figure 6: Effectiveness Analysis Scenario II: (a) Precision and (b) Recall for detecting clusters

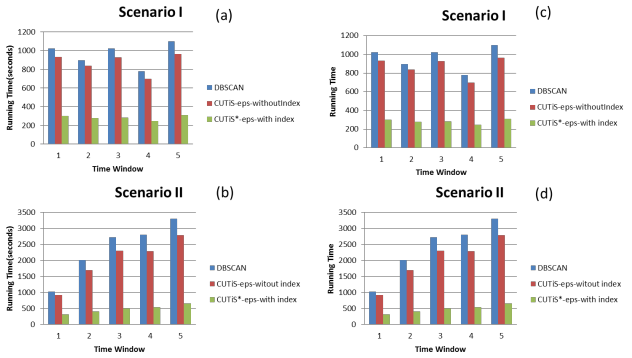


Figure 7: (a) and (b): Efficiency Analysis (Improved Running Time by using our proposed index); (c) and (d): Efficiency Analysis of Clustering Approaches

spatial position at each timestamp and some of them incrementally maintain the clusters as time goes by. Papers [14, 16] also predict when a moving object will leave or join a cluster (on the other hand, CUTiS* can observe the real displacement behavior of objects). Furthermore, [14] does not apply density-based clustering and [15] considers only update on moving object position as time goes by.

Movement Pattern Discovery. A very related topic to this study is to discover collective patterns among moving objects, as flock, swarm, convoy, herds, gathering, among others [17]. The published approaches to find these patterns are very sensitive to specific parameters. Hence, these methods cannot be directly applicable to our problem since they do not report sub-trajectory clusters and their evolution.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented CUTiS* which tracks moving objects, incrementally maintains sub-trajectory clusters using trajectory data streams and captures the sub-trajectory clusters evolution from time to time. We also proposed an indexing technique for trajectory data based on the Hilbert curve, to improve performance. Our experiments were conducted with a real dataset and compared the efficiency and effectiveness of CUTiS* with that of our competitors and the performance improvement obtained by using the index. In the future, we aim at extending our method to find other mobility patterns (including predict future clusters events) and to explore the application of trajectory segmentation techniques without losing quality in the clusters results.

Acknowledgement. Research supported by CNPq in Brazil and performed while the first author was visiting David Laboratory in UVSQ, France.

7. REFERENCES

[1] T. Coelho da Silva, K. Zeitouni, and J. de Macêdo. Online clustering of trajectory data streams. In *MDM*, 2016.

[2] T. Coelho da Silva, K. Zeitouni, J. de Macêdo, and M. A. Casanova. On-line mobility pattern discovering using trajectory data. In *EDBT*, pages 682–683, 2016.

[3] T. Coelho da Silva, K. Zeitouni, J. de Macêdo, and M. A. Casanova. A framework for online mobility pattern discovery from trajectory data stream. In *MDM*, 2016.

[4] A. N. Araujo, T. Coelho da Silva, V. de Farias, J. de Macêdo, and J. Machado. G2P: A partitioning approach for processing dbscan with mapreduce. In *W2GIS*, pages 191–202. 2015.

[5] H. V. Jagadish. Linear clustering of objects with multiple attributes. In *SIGMOD*, pages 332–342, 1990.

[6] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, pages 593–604, 2007.

[7] M. Nanni and D. Pedreschi. Time-focused clustering of trajectories of moving objects. *JIIS*, pages 267–289, 2006.

[8] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering uncertain trajectories. *KAIS*, pages 117–147, 2011.

[9] X. Li, J. Han, J. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *SSTD*, pages 441–459. 2007.

[10] B. Han, L. Liu, and E. Omiecinski. Neat: Road network aware trajectory clustering. In *ICDCS*, pages 142–151, 2012.

[11] Z. Li, J. Lee, X. Li, and J. Han. Incremental clustering for trajectories. In *DASFAA*, pages 32–46, 2010.

[12] Y. Yu, Q. Wang, X. Wang, H. Wang, and J. He. Online clustering for trajectory data stream of moving objects. *ComSIS*, pages 1293–1317, 2013.

[13] J. Mao, Q. Song, C. Jin, Z. Zhang, and A. Zhou. Tscluwin: trajectory stream clustering over sliding window. In *DASFAA*, pages 133–148, 2016.

[14] C. S. Jensen, D. Lin, and Beng-Chin Ooi. Continuous clustering of moving objects. *TKDE*, 2007.

[15] L. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C. Hung, and W. Peng. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012.

[16] X. Li, V. Ceikute, C. S. Jensen, and K. Tan. Effective online group discovery in trajectory databases. *TKDE*, pages 2752–2766, 2013.

[17] Y. Zheng. Trajectory data mining: an overview. *TIST*, 2015.