Time-aware Sub-Trajectory Clustering in Hermes@PostgreSQL

Panagiotis Tampakis [1], Nikos Pelekis [2],
Yannis Theodoridis [6]
Data Science Lab
University of Piraeus
Piraeus, Greece
[1,2,6]{ptampak|npelekis|ytheod}@unipi.gr

Natalia Andrienko [3],
Gennady Andrienko [4], Georg Fuchs [5]
Fraunhofer Institute IAIS
Sankt-Augustin, Germany
[3,4,5]{natalia.andrienko|gennady.andrienko|georg.fuchs}@iais.fraunhofer.de

***Abstract*—In this paper, we present an efficient in-DBMS framework for progressive time-aware sub-trajectory cluster analysis. In particular, we address two variants of the problem: (a) spatiotemporal sub-trajectory clustering and (b) index-based time-aware clustering at querying environment. Our approach for (a) relies on a two-phase process: a voting-and-segmentation phase followed by a sampling-and-clustering phase. Regarding (b), we organize data into partitions that correspond to groups of sub-trajectories, which are incrementally maintained in a hierarchical structure. Both approaches have been implemented in Hermes@PostgreSQL, a real Moving Object Database engine built on top of PostgreSQL, enabling users to perform progressive cluster analysis via simple SQL. The framework is also extended with a Visual Analytics (VA) tool to facilitate real world analysis.**

I. Introduction

Knowledge discovery in mobility data [11] exposes patterns of moving objects useful in several fields, such as transportation, climatology, zoology, mobile social networks. Mobility (mostly GPS-based) data capturing results in trajectories of moving objects stored in Moving Object Databases (MOD), calls for novel, sophisticated, efficient, comprehension and analysis of mobility.

In the literature of trajectory-based data mining [13], one can identify several types of mining models used to describe various collective behavioral patterns. Focusing on trajectory clustering, the majority of related work proposes a variety of distance functions, utilized by well-known clustering algorithms to identify collective behavior among entire trajectories. In a parallel line of research most related to the current approach, researchers aim to discover local patterns in MOD, i.e. co-movement patterns that are alive only for a portion of moving objects' lifespan, such as moving clusters, flocks, convoys, swarms, platoons and other patterns [13]. Other techniques, such as TRACLUS [5], simplify and partition the given trajectories and then apply density-based clustering, focusing on the spatial and ignoring the temporal dimension.

Exploration of clustering results is often supported by interactive Visual Analytics (VA) tools, as in the examples illustrated in Fig. 1. The dataset employed for this example is a real MOD consisting of aircrafts approaching airports of the London metropolitan area. However, it is straightforward to employ datasets from other domains, such as maritime or urban traffic movement. For instance, the cluster affiliation of trajectory segments is represented on a map display by color coding. The user can interactively select which clusters to show or hide, in order to be able to examine selected clusters in detail. The existence times of the clusters and the changes of their cardinality over time can be explored using a time histogram, in which bars are divided into segments painted in the same colors as the cluster members in the map. The 3D shapes of the cluster members can be seen in a 3D display. In general, VA systems provide a number of visual and interactive techniques designed to support mobility data exploration and analysis [1].

Towards the goal of interactive mobility data exploration and analysis, our motivation in this work is to demonstrate how a MOD engine built on top of extensible DBMS can efficiently incorporate two sub-trajectory clustering algorithms proposed recently, namely Sampling-based Trajectory Clustering (S^2T-Clustering) [9] and Query-based Trajectory Clustering (QuT-Clustering) [10]. Interestingly, both algorithms operate on the entire spatio-temporal domain, by overpassing on the one hand some limitations of the state-of-the-art TRACLUS framework, while on the other hand eliminating hard-to-tune parameters as those introduced in the co-movement patterns approaches. Most importantly, with our approach we demonstrate the feasibility of progressive time-aware analytics, in terms that we allow a data analyst to select different time periods to perform his/her analysis, without being obliged to apply from scratch costly preprocessing or iterative clustering procedures.

The practical contribution of this work is that we present a framework that fulfils two significant specifications: (a) implements efficient and scalable solutions for sub-trajectory clustering that (b) operate on a real-world DBMS rather than being ad hoc implementations. The in-DBMS implementation of our methods is performed in Hermes@PostgreSQL[1], our open source Moving Object Database (MOD) engine built on

[1] Available for download at www.datastories.org/hermes.

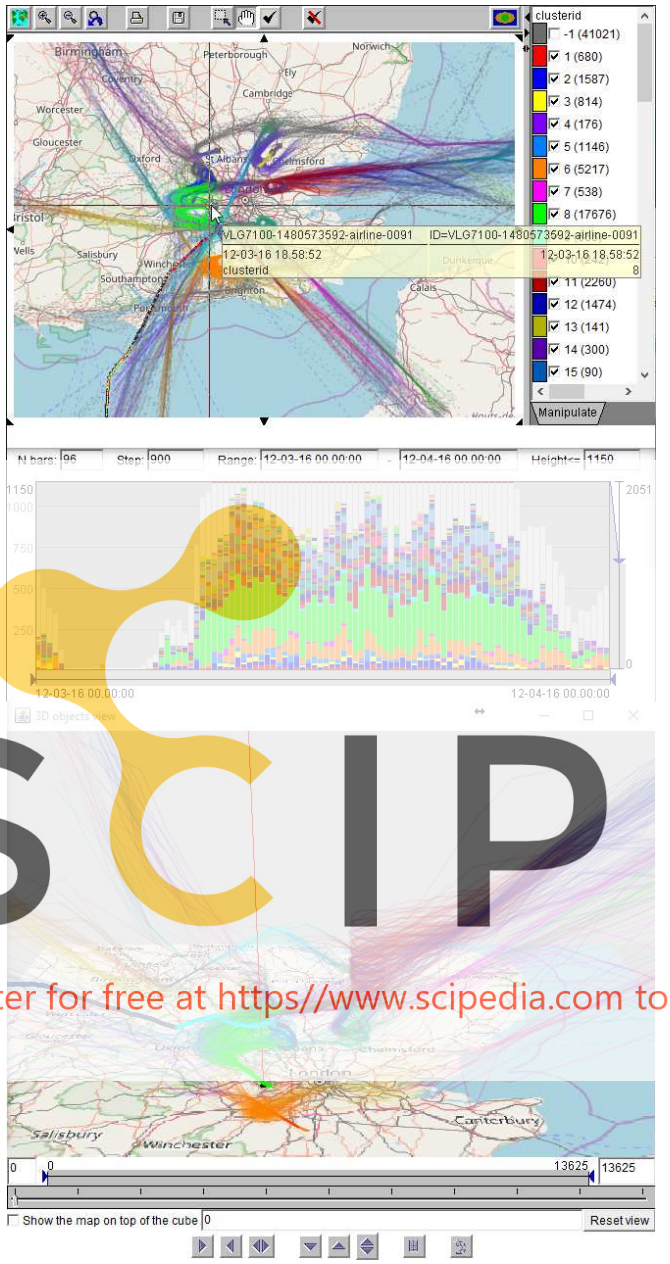

Figure 1. Interactive visual exploration of clustering results: map display of clusters (top); evolution of cardinality of clusters over time (middle); 3D shapes of cluster members (bottom).

top of PostgreSQL, by using GiST [3] extensibility interface provided by PostgreSQL. To our knowledge, it is the first time in the literature that GiST is used to index trajectory-based mobility data for the above purposes. More specifically, GiST is utilized in order to build a 3D-RTree index tailored for trajectories. Therefore, we argue that this is a step towards bridging the gap between MOD management and mobility data mining, as state-of-art frameworks [6][12][1] could make use of the efficiency and the advantage of our approach to execute in-DBMS (sub-)trajectory clustering via simple SQL queries.

II. Major Modules and System Architecture

In this section, we present the details of the two major modules of our approach, namely S^{2}T-Clustering and QuT-Clustering. Then, we provide the overall architecture of our system.

A. S^{2}T-Clustering

In general, the objective of sub-trajectory clustering is to partition trajectories into sub-trajectories and then form groups of similar ones, while at the same time separate those (called outliers) that fit into no group. S^{2}T-Clustering [9] is a state-of-the-art algorithm consists of two phases: during the first phase, a Neighborhood-aware Trajectory Segmentation (NaTS) method is applied over trajectories, thus splitting them in sub-trajectories; during the second phase, Sampling, Clustering and Outlier (SaCO) detection steps are performed in order to provide the final result. NaTS relies on a voting and segmentation process that detects homogenized sub-trajectories in the MOD w.r.t. how many other objects move in their neighborhood, while SaCO selects the most representative ones to serve as the seeds of the clusters, around which the clusters are formed (also, the outliers are isolated).

In more detail, during the adopted **voting** process each 3D trajectory segment of a given trajectory is voted by other trajectories w.r.t. their mutual distance. The voting received by each segment is a value ranging from 0 to N (N being the cardinality of the MOD) that has the physical meaning of how many trajectories co-move with that trajectory for a certain period of time. After the voting process takes place, the trajectory **segmentation** process follows. The goal of this step is to partition each trajectory into sub-trajectories having *homogenous representativeness*, irrespectively of their shape complexity. However, the goal of sub-trajectory clustering is to partition the entire dataset into groups (clusters) and to detect the outliers among the sub-trajectories identified by the trajectory segmentation step. Therefore, in our proposal, we first select the appropriate **sampling** set S and then, we tackle the problem of clustering according to the following idea: *each sub-trajectory in the sampling set is considered to be a cluster representative*. So, the sampling set should contain highly voted trajectories of the MOD which, at the same time, would cover the 3D space occupied by the entire dataset as much as possible. Then, the **clustering** is done building the clusters "around" those representatives. For more details about S^{2}T-Clustering, please refer to [9].

B. QuT-Clustering

In summary, QuT-Clustering [10] relies upon a hierarchical structure, called ReTraTree (for Representative Trajectory Tree) that effectively indexes a MOD for sub-trajectory clustering purposes. ReTraTree consists of four levels: the first two levels operate on the temporal dimension, the third level builds clusters upon the spatio-temporal characteristics of the trajectories, and the fourth level is the actual data storage along with the corresponding indexes (3D-RTree) for effective retrieval.

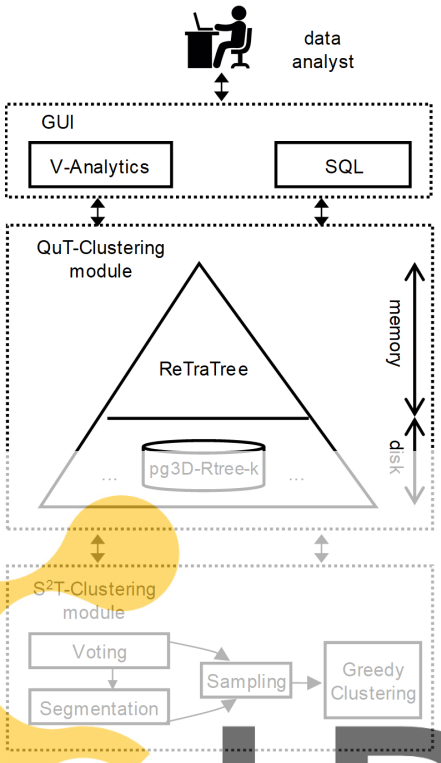

Figure 2. Architecture of the time-aware sub-trajectory clustering module implemented in Hermes@PostgreSQL

Given a MOD indexed according to ReTraTree structure and a temporal period W of interest, QuT-Clustering efficiently retrieves the subset of the MOD, actually the clusters and outliers at sub-trajectory level, that temporally intersect W. The structure of the QuT-Clustering query in SQL follows:

```
SELECT QUT(D, W_i, W_e, τ, δ, t, d, γ);
```

where D is the dataset name, W_i and W_e are the initial and the ending time of the temporal period W, and τ, δ, t, d, γ correspond to the respective parameters of the QuT-Clustering algorithm [10]. For more details about ReTraTree and the QuT-Clustering algorithm, please refer to [10].

C. System Architecture

The architecture of our framework is illustrated in Fig. 2. As expected, the core of the framework is the ReTraTree. The trajectories composing the MOD are partitioned according to the in-memory part of the structure and stored on disk-based partitions. The trajectories assigned to an existing representative trajectory are archived on disk in dedicated R-tree indexed partitions (called 'pg3D-Rtree-k' in Fig. 2). On the other hand, outlier trajectories are organized on disk in a separate partition. When the size of a partition exceeds a pre-defined threshold, S^{2}T-Clustering takes action: it applies Voting among trajectories, with the voting results indicating the Segmentation of trajectories into sub-trajectories that should take place. The resulting sub-trajectories along with their voting descriptors feed the Sampling module that selects new representatives, which are then back-propagated to the in-memory part of ReTraTree. The new representative trajectories as well as the raw sub-trajectories form the input of the GreedyClustering module: if a sub-trajectory is clustered around a new representative, it is archived in its appropriate partition on disk; otherwise, it is considered outlier and is re-inserted to ReTraTree, as it may now be accommodated in the index. Note that our implementation is completely independent from PostGIS. This implies that the underlying R-tree index, coined pg3D-Rtree in Fig. 2, has also been implemented from scratch on top of GiST.

Having this functionality in hand, the data analyst is able to perform interactive clustering analysis, by providing different values of W as input, through either the SQL interface of Hermes@PostgreSQL [11] or the incorporated V-Analytics tool [1].

III. About The Demonstration

Throughout the demonstration, users will be able to test the system using a real dataset of moving objects. The users will have the chance to catch a glimpse "under the hood", experiment and visualize the results of some state of the art clustering techniques, such as [9] and [10]. More specifically, the demonstration captures the following phases - scenarios:

Preparatory phase (background knowledge): Initially, the user has the opportunity to comprehend the internals of our implementation and API, which exploits on the extensibility interface given by PostgreSQL. We show off the datatypes and operands resulting in Hermes@PostgreSQL MOD engine. In addition, we demonstrate how the user can use our SQL API to run all legacy operands, and even more interestingly, tools on the two sub-trajectory clustering approaches, allowing orders of magnitude speedup in comparison to corresponding PostgreSQL functions [9].

In action phase – scenario 1: Having gained the necessary background knowledge, the user experiences a progressive clustering scenario based on the S^{2}T-Clustering algorithm [9] as well as related methods, such as T-OPTICS [7], TRACLUS [5] and Convoys [4], and VA methods that aim at the analysis of the discovered patterns. For instance, Fig. 3 presents an example of results of two runs of S^{2}T-Clustering, for comparison purposes. The cluster representatives from the two runs are viewed in a 3D display. The user can either put both sets of results in the same 3D display or create two 3D displays, each showing one set of results. In the first case, the user can interactively switch on and off the visibility of each set of results. The same can be done with a map display. Moreover, the user experiences in discovering and visualizing other interesting patterns, such as the holding patterns typically performed by aircrafts as they approach to their destination, in our case London airports (as it is illustrated in Fig. 4).

In action phase – scenario 2: In turn, we present a progressive clustering scenario, this time focusing on the temporal dimension and highlighting the QuT-Clustering

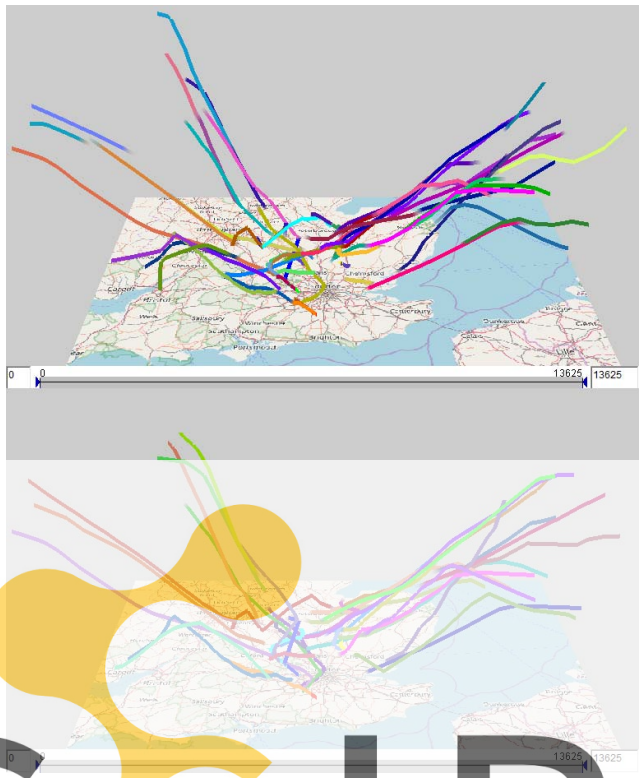

Figure 3. Time-aware sub-trajectory clustering in action: cluster representatives from two different runs of S^{2}T-Clustering are visually compared by means of a 3D display.

Figure 4. Time-aware sub-trajectory clustering in action (cont.): holding patterns performed by aircrafts are discovered and visualized.

functionality. The goal of this scenario is twofold: first, we demonstrate via the SQL API the efficiency speedup of performing the clustering task for varying time periods W. We compare QuT-Clustering with the alternative approach that consists of (i) extracting the relevant records using a temporal range query, (ii) creating an R-tree index on the result of the query, and (iii) applying clustering (S^{2}T-Clustering, in our case). Second, we follow a similar approach, but this time we use the V-Analytics component of our framework in order to comprehend the evolvement of the sub-trajectory patterns with increasing time periods W. In detail, by setting small value of W we focus on the landing phase of aircrafts and visualize the discovered clusters (recall, e.g. Fig. 3); then, we increase the value of W to the past in order to realize the evolution of patterns as aircrafts pass from the cruising to the landing phase.

For deeper comprehension of both progressive analysis scenarios (S^{2}T-Clustering and QuT-Clustering) to be demonstrated, two related videos are available at Hermes@PostgreSQL demo web page[2].

[2] URL: www.datastories.org/hermes/demo.

ACKNOWLEDGMENT

This work was partially supported by projects datACRON (grant agreement No 687591), Track&Know (grant agreement No 780754) and MASTER (Marie Sklowdoska-Curie agreement N. 777695), which have received funding from the EU Horizon 2020 R&I Programme.

REFERENCES

[1] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel. Visual Analytics of Movement. Springer, 2013.

[2] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. 2011. Unveiling the Complexity of Human Mobility by Querying and Mining Massive Trajectory Data. *The VLDB Journal — The International Journal on Very Large Data Bases*, 20(5): 695-719.

[3] J. Hellerstein, J. Naughton and A. Pfeffer, 1995. Generalized Search Trees for Database Systems. In *Proceedings of VLDB*.

[4] H. Jeung, M.L. Yiu, X. Zhou, C.S. Jensen, and H.T. Shen. 2008. Discovery of Convoys in Trajectory Databases. In *Proceedings of the VLDB Endowment*, pages 1068-1080.

[5] J.G. Lee, J. Han, and K.Y. Whang. 2007. Trajectory Clustering: A Partition-and-Group Framework. In *Proceedings of SIGMOD*.

[6] Z. Li, M. Ji, J-G. Lee, L-A. Tang, Y. Yu, J. Han, and R. Kays. 2010. MoveMine: Mining Moving Object Databases. In *Proceedings of SIGMOD*.

[7] M. Nanni and D. Pedreschi. 2006. Time-focused Clustering of Trajectories of Moving Objects. *Journal of Intelligent Information Systems*, 27(3):267-289.

[8] C. Panagiotakis, N. Pelekis, I. Kopanakis, E. Ramasso, Y. Theodoridis. 2012. Segmentation and Sampling of Moving Object Trajectories based on Representativeness, IEEE Transactions on Knowledge and Data Engineering, 24(7):1328-1343.

[9] N. Pelekis, P. Tampakis, M. Vodas, C. Panagiotakis, Y. Theodoridis. 2017. In-DBMS Sampling-based Sub-trajectory Clustering, In *Proceedings of EDBT*.

[10] N. Pelekis, P. Tampakis, M. Vodas, C. Doulkeridis Y. Theodoridis. 2017. On Temporal-Constrained Sub-Trajectory Cluster Analysis, *Data Mining and Knowledge Discovery*, 31(5):1294-1330.

[11] N. Pelekis, and Y. Theodoridis. 2014. *Mobility Data Management and Exploration.* Springer.

[12] F. Wu, T.K.H. Lei, Z. Li, and J. Han. 2014. MoveMine 2.0: Mining Object Relationships from Movement Data. In *Proceeding of the VLDB Endowment*, pages 1613-1616.

[13] Y. Zheng. 2015. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3), article no. 29.