Visualization of Range-Constrained Optimal Density Clustering of Trajectories

Muhammed Mas-Ud Hussain[1(✉)], Goce Trajcevski[1], Kazi Ashik Islam[2], and Mohammed Eunus Ali[2]

[1] Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA
mas-ud@u.northwestern.edu, goce@eecs.northwestern.edu
[2] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
1205007.kai@ugrad.cse.buet.ac.bd, eunus@cse.buet.ac.bd

Abstract. We present a system for efficient detection, continuous maintenance and visualization of range-constrained optimal density clusters of moving objects trajectories, a.k.a. Continuous Maximizing Range Sum (Co-MaxRS) queries. Co-MaxRS is useful in any domain involving continuous detection of "most interesting" regions involving mobile entities (e.g., traffic monitoring, environmental tracking, etc.). Traditional MaxRS finds a location of a given rectangle R which maximizes the sum of the weighted-points (objects) in its interior. Since moving objects continuously change their locations, the MaxRS at a particular time instant need not be a solution at another time instant. Our system solves two important problems: (1) Efficiently computing Co-MaxRS answer-set; and (2) Visualizing the results. This demo will present the implementation of our efficient pruning schemes and compact data structures, and illustrate the end-user tools for specifying the parameters and selecting datasets for Co-MaxRS, along with visualization of the optimal locations.

1 Introduction

Advances in miniaturization of GPS and sensor-equipped (position-aware) devices, along with progresses in networking and communications, have resulted in the generation of large quantities of (*location, time*) data, (O(Exabyte)) [1]. Movement of people, animals, vehicles, birds, etc. are continuously captured by GPS loggers, and as such, the efficient management of mobility data is at the core of many applications of high societal relevance. Often, such data are being scrutinized by clustering [2,3], mining, information retrieval [4], and visualization techniques [5,6] to detect patterns of interest among the trajectories.

This work explores efficient processing and visualization of the spatio-temporal extension of a particular type of spatial query – the, so called, Maximizing Range-Sum (MaxRS) query, which can be described as follows. Given

M. Mas-Ud Hussain and G. Trajcevski—Research supported by NSF grants III 1213038 and CNS 1646107, ONR grant N00014-14-10215 and HERE grant 30046005.

M. Gertz et al. (Eds.): SSTD 2017, LNCS 10411, pp. 427–432, 2017.
DOI: 10.1007/978-3-319-64367-0_29

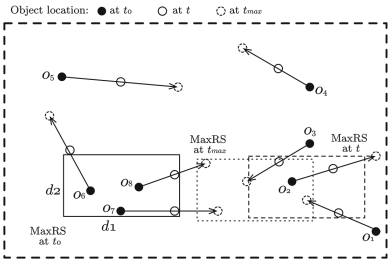


Fig. 1. MaxRS at three different time instants ($t_0 < t < t_{max}$).

a set O of weighted objects and a rectangle R (dimension $d_1 \times d_2$), MaxRS retrieves a location of R that maximizes the sum of the weights of all the objects in its interior. MaxRS, and its variants for scalability, approximate solutions, etc., have been addressed in spatial settings [7,8]. However, many applications involve moving objects and the *continuous* variant of MaxRS is needed – Fig. 1 shows how the locations of the instantaneous MaxRS solutions vary over time for mobile objects.

Co-MaxRS detects and maintains "most interesting" regions over moving objects trajectories, i.e., highest-density clusters given the range R and time-period $T = [t_0, t_{max}]$, which is essential for: environmental tracking (e.g., optimizing a range-bounded continuous monitoring of a herd of animals); – traffic monitoring (e.g., detecting ranges with densest traffic between noon and 6PM); – video-games (e.g., determining a position of maximal coverage in dynamic scenarios). Works on trajectories clustering [2,3] did not track the most dense (based on user-defined criteria) clusters over time. Moreover, proper visualization of Co-MaxRS answer-set for different time-periods and ranges would be handy for focusing on specific optimal regions when analyzing large trajectory data. Recent works on continuous visualization of spatial data have different settings from us: – [5] enables visualizing DBSCAN-based clusters for trajectories (no fixed range or optimal clustering considered) and [6] deals with static spatial events of mobile entities (i.e., does not consider the trajectories themselves).

We provide a system for: (1) Efficiently maintaining Co-MaxRS answer-set for a given range and time-period; and (2) Providing a user-friendly GUI to enable visualization of the answer-set. In [9], we introduced useful pruning schemes and compact data structures (plus TPR*-tree indexing) to enable cost-effective processing of Co-MaxRS queries. This demo paper presents the system which implements the techniques proposed in [9] over several real-world datasets obtained from CRAWDAD (http://crawdad.org) from various domains, along with the GUI for users to specify the desired parameters and even provide their own dataset, and modules for visualizing the Co-MaxRS answer-set.

2 System Design and Demo Details

We now describe the main components of the system architecture and how they interact with each other, and proceed with details of the demo.

Software Architecture: Our final system is a web-based application with interactive and user-friendly interface for both PC and mobile devices, which is implemented using HTML, CSS, and JavaScript (Node.js was used for the server-side programming). We employ *Responsive Web Design* principles in building the website by using CSS media queries, @media rules, and fluid grids—thus, making it suitable to work on every device and screen size. The core Co-MaxRS algorithm is implemented using C++. The software architecture of our system, illustrated in Fig. 2, is organized in the following main categories:

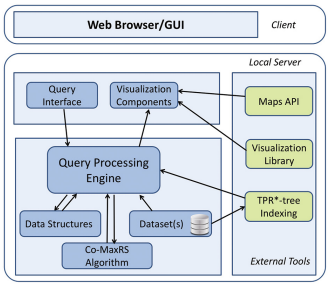


Fig. 2. Software architecture.

- *Query Interface:* The users can select each of the required parameters using the interface by specifying: (1) $\mathbb{F}$ – the query area (depends on the underlying dataset), set dynamically via the zoom-in (+) or out (−) tool; (2) R – specifying d_1 and d_2 values; and (3) T – the time period, specifying t_0 and t_{max}. Users can browse through solutions for each T by sliding the time bar. Additionally, a user will also be able to upload their own dataset, as long as it follows a prefixed format – plain file containing tuples of (*object-id, trajectory-id, latitude, longitude, time, weight* (*optional, default=1*)).
- *Visualization Components:* We used Google Maps JavaScript API [10] to display a map with respective pins for each object at a particular time instant and the current MaxRS solution. These pins and results will change accordingly as the user drags the time bar. Also, we used various JavaScript visualization tools (such as [11]) to enable a different view of the result – trajectories of the Co-MaxRS solutions in 3D spatio-temporal settings.
- *Co-MaxRS Algorithm:* Even for small object movements, the optimal location of the query rectangle can change while objects participating in the MaxRS solution stay the same. Instead of maintaining a centroid-location (equivalently, a region) as a Co-MaxRS solution, we maintain a list of objects that are located in the interior of the optimal rectangle placement – the

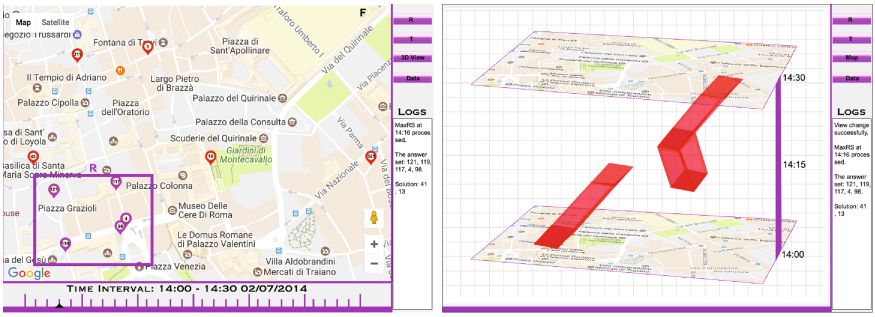
Co-MaxRS answer-set becoming a sequence (*list-of-objects, time-period*). For the example in Fig. 1, Co-MaxRS answer-set is: $\{((o_6, o_7, o_8), [t_0, t_1)), ((o_1, o_2, o_3), [t, t_{max})), ((o_1, o_3, o_7, o_8), [t_{max}, t_{max}))\}$. We identified criteria (i.e., critical times) when a particular MaxRS solution is no longer valid, or a new MaxRS solution emerges. The basic algorithm uses KDS (*Kinetic Data Structures*) – maintaining a priority queue of the critical events and their occurrence time, and recomputing MaxRS solutions at each event in order. Recomputing MaxRS is costly, so we devised efficient pruning schemes to: (1) Eliminate the recomputation altogether at certain qualifying critical events; and (2) Reduce the number of objects that need to be considered when recomputation is unavoidable.

- *Data Structures and Indexing:* We maintain a list for storing each object $o_i \in O$, with its current trajectory, weight, and other necessary information. KDS maintains an event queue, where the events are sorted according to the time-value. Each critical event consists of the related objects, and the occurrence time. Additionally, we also maintained an adjacency matrix to track locality of objects – which is important for smooth processing of our pruning schemes. Moreover, to ensure faster processing over large datasets, we used an existing spatio-temporal data indexing scheme, TPR*-tree (via a C++ library) [12].
- *Datasets:* To demonstrate the benefits of our system in different domains, we use several datasets: (1) GPS traces from 500 taxis collected over 30 days in San Francisco Bay area (http://crawdad.org/epfl/mobility/); (2) (*location, time*) information for 370 taxis within Rome for 1 month with sample-period of 7 s (http://crawdad.org/roma/taxi/); (3) Human mobility data, in [13], where researchers at Microsoft collected data from 182 users in a period of over five years, with 17,621 trajectories; and (4) A small animal movement dataset from (http://crawdad.org/princeton/zebranet/). While the first two datasets are great for demonstrating scalability and traffic-monitoring aspect of our system, the latter two can be used to show applications in human mobility tracing and animal tracking processes. Although all of these datasets had different formats, we converted them into the same format – tuples of (*object-id, trajectory-id, latitude, longitude, time, weight*) values. This enabled our system to handle similarly formatted user-provided data as well.

Demo Specifications: The setup of our demo will consist of a laptop running the web-based application via a web browser. The application is hosted on our server operating at Northwestern University, and will be accessed via a public URL. The demonstration will have three main parts with the following steps:

P1: The first part will have following three main phases:

Phase 1: Specification of the required parameters in the GUI (cf. Fig. 3). This phase will show: (a) Selection of appropriate R, $\mathbb{F}$, and T; and (b) Relation between R and $\mathbb{F}$, i.e., how our system dynamically changes R proportionally to $\mathbb{F}$ given an initial value. $\mathbb{F}$ can be dynamically set by the user ($\pm$ tool).



(a) (b)

Fig. 3. (a) The main GUI with map and time slider; (b) Answer-set 3D view over T.

Phase 2: Visualization of the Co-MaxRS result at a certain point of time t using the user-provided parameters over a map (constrained to $\mathbb{F}$). The locations (linearly interpolated at t) of the objects will be displayed via pins on the map, and the solution, i.e., an optimal placement of R and the objects in its interior will be distinctly marked (see Fig. 3a).

Phase 3: Visualization of the whole answer-set for the time-period T in spatio-temporal (3D) settings (cf. Fig. 3b). Users will be able to analyze how the trajectories of the optimal clusters are evolving over space and time (parallelopipeds).

P2: In the second and most important part of the demo, we will exhibit the benefits of using this tool in analyzing large trajectories data. This part will show: (a) Selection of the area of focus ($\mathbb{F}$) by zooming-in or out; (b) Change the value of t by sliding through per unit time (depends on the data set) within T. We will demonstrate how a user can start from a large $\mathbb{F}$ and relatively larger R, and then continuously refine their region of interest.

P3: The last part of the demo will quickly show steps of **P1** and **P2** for different datasets, emphasizing how the tool can be useful in different domains. Additionally, the steps of uploading a custom dataset will be demonstrated.

References

1. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big data: The next frontier for innovation, competition, and productivity
2. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. J. Intell. Inf. Syst. **27**(3), 267–289 (2006)
3. Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D., Giannotti, F.: Interactive visual clustering of large collections of trajectories. In: IEEE Symposium on Visual Analytics Science and Technology (2009)
4. Zheng, Y.: Trajectory data mining: an overview. ACM Trans. Intell. Syst. Technol. (TIST) **6**(3), 29 (2015)

5. Shen, Y., Zhao, L., Fan, J.: Analysis and visualization for hot spot based route recommendation using short-dated taxi GPS traces. Information **6**(2), 134–151 (2015)
6. Andrienko, N., Andrienko, G., Fuchs, G., Rinzivillo, S., Betz, H.D.: Detection, tracking, and visualization of spatial event clusters for real time monitoring. In: IEEE DSAA (2015)
7. Choi, D.W., Chung, C.W., Tao, Y.: Maximizing range sum in external memory. ACM Trans. Database Syst. **39**(3), 21:1–21:44 (2014)
8. Nandy, S.C., Bhattacharya, B.B.: A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids. Comput. Math. Appl. **29**(8), 45–61 (1995)
9. Hussain, M.M., Islam, K.A., Trajcevski, G., Ali, M.E.: Towards efficient maintenance of continuous MaxRS query for trajectories. In: 20th EDBT (2017)
10. Google Maps API. https://developers.google.com/maps/
11. Vis.js: JavaScript visualization library. https://visjs.org/
12. Tao, Y., Papadias, D., Sun, J.: The TPR*-tree: an optimized spatio-temporal access method for predictive queries. In: VLDB (2003)
13. Zheng, Y., Zhang, L., Xie, X., Ma, W.Y.: Mining interesting locations and travel sequences from GPS trajectories. In: ACM World Wide Web (2009)