GEDetector: Early Detection of Gathering Events Based on Cluster Containment Join in Trajectory Streams

Bin Zhao
School of Computer Science and Technology
Nanjing Normal University, China
zhaobin@njnu.edu.cn

Genlin Ji*
School of Computer Science and Technology
Nanjing Normal University, China
glji@njnu.edu.cn

Yu Yang
School of Computer Science and Technology
Nanjing Normal University, China
162202020@njnu.edu.cn

Zhaoyuan Yu
Key Laboratory of Virtual Geographic Environment Ministry of Education
Nanjing Normal University, China
yuzhaoyuan@njnu.edu.cn

Xintao Liu
Department of Land Surveying and Geo-Informatics
The Hong Kong Polytechnic University, Hong Kong
xintao.liu@polyu.edu.hk

Ningfang Mi
Electrical and Computer Engineering Department
Northeastern University, Boston, MA USA
ningfang@ece.neu.edu

ABSTRACT

Existing trajectory patterns, such as flock, convoy, swarm, and gathering, are to detect moving clusters staying or travelling together for a certain time period. But these patterns model group movement behaviors after moving objects' gathering together. This may result in loosing golden opportunities to detect emergency incidents earlier, such as traffic congestion and serious stampedes. In this work, we propose a novel group pattern, called converging, which can model converging behaviors of moving objects. As a proof-of-concept, we implemented a visual analytic system GEDetector based on trajectory streams to detect gathering events as early as possible. A user-friendly interface is designed to help users gain insights into gathering events from spatial and temporal aspects. Finally, we demonstrate the effectiveness and efficiency of our system by using a real world dataset.

1 INTRODUCTION

The existing group pattern mining methods, such as flock [1], convoy [2], swarm [3], travelling companion [4], and gathering [5], are to discover a group of objects that stay or travel together for a certain time period [6]. Analysis tasks based on group patterns can be used for enormous applications, including transportation optimization, public security, advertisement delivery, travel recommendation, and animal movement study and so on [7]. In this work, we develop a visual analytic system GEDetector to detect gathering events in trajectory streams.

Previous studies on group patterns are unsuitable for detecting gathering events despite of wide applications of group patterns. It is worth pointing out that, most of group patterns can capture group movement behaviors only after moving objects gather together. For example, convoys require that each group of objects travel together during the whole pattern lifetime. Gatherings also require that all participators stay together at the most timestamps in the pattern lifetime. However, in real life, people are more interesting in modelling and identifying group converging behaviors before the gathering events actually happen, since this helps to proactively report and handle the coming public incidents, such as traffic congestion and serious stampedes.

However, the efficient discovery of convergings in trajectory streams is a challenging task due to two main reasons: (1) the existing group patterns cannot capture intuitively converging behaviors of gathering events; (2) the discovery of convergings may face huge search space and incur high cost. In this work, we propose a novel group pattern, called converging, which can model the converging behaviors of moving objects and detect gathering events in trajectory streams. Furthermore, we propose a converging pattern mining method based on cluster containment join (CCJ), which utilizes a signature quad-tree based index, called SQTI, to organize clusters hierarchically and spatially. The SQTI based mining algorithm enables us to rapidly reduce search space and efficiently identify interesting and useful converging patterns.

In this demo, we present a novel Gathering Event Detection system, named GEDetector, which is designed to achieve an early detection of gathering events through mining converging patterns in trajectory streams. The GEDetector has been deployed on a virtual machine of Alibaba Cloud, which can be accessed at *http://103.242.175.191: 5456/gedetector*.

2 PROBLEM FORMULATION

The identifying characteristic of a gathering event is a *converging*, which is a group of moving objects gathering from different directions for at least k_t timestamps. It is easy to see that, converging patterns focus on the earlier stages of gathering events compared to other existing patterns. To accurately model converging behaviors, we introduce *participators*, which are used to indicate the objects appearing in at least k_p consecutive clusters of this converging, and require that a converging should contain at least k_m participators.

Now we use Fig. 1 as an example to illustrate the converging pattern, and let $k_t = 2$, $k_p = 2$, $k_m = 4$, in which there are six moving objects joining a gathering event from different directions and forming one cluster. Additionally, there are two clusters (i.e., c_1 & c_2) that are gathered at t_2 and later join the cluster c_3 at t_3. Such set containment between two clusters is called *cluster containment relation*, denoted by $\subseteq_c$. Thus, we have $c_1 \subseteq_c c_3$ & $c_2 \subseteq_c c_3$ in Fig. 1(a). By joining these two cluster containment relations, we can construct a containment tree to model converging behaviors of moving objects, as shown in Fig.

1(b). To realistically model converging behaviors, a converging requires to have enough participators who appear in the most snapshot clusters during the event lifetime. As shown in Fig. 1(c), the group of participators $\langle o_1, o_2, o_3, o_4, o_5 \rangle$ satisfies the above requirements because they appear in two clusters (i.e., c_1 & c_2) at time t_2 and t_3. But, the group excludes o_6, since o_6 only appears in one snapshot cluster, which is less than the given threshold $k_p = 2$.

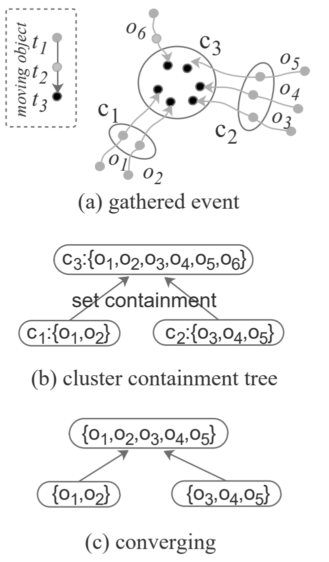


Figure 1: Example of a converging

3 FRAMEWORK OF GEDETECTOR

Fig. 2 shows the framework of GEDetector, which consists of three major modules: (1) **snapshot cluster discovery (SCD)**: we perform density-based clustering on the moving objects to discover a set of snapshot clusters at each timestamp; (2) **cluster containment join (CCJ)**: we perform a cluster containment join on any two snapshot cluster sets at consecutive timestamps, and return a collection of cluster containment relations; and (3) **converging detection (CD)**: we construct cluster containment trees through joining all cluster containment relations, and further derive the qualifying results from the candidates of cluster containment trees according to the aforementioned requirements of converging concepts.

The detection procedure of GEDetector is as follows. The initial input of GEDetector is a trajectory data stream that is periodically collected at timestamp t_i. When the latest batch of trajectory data is appended to the trajectory database, GEDetector will trigger the detection process in an incremental manner, as shown in Fig. 2. At t_i, the snapshot clusters are first discovered from the new trajectory data by clustering algorithms. Then, the CCJ is performed between the snapshot cluster sets of t_{i-1} and t_i. Correspondingly, the new convergings may be generated, and some existing convergings may be also updated. The procedures of the first two modules (i.e., SCD & CCJ) can be repeated until no new trajectory data is given. Finally, the qualified converging patterns are derived by the third module (CD).

The efficient discovery of convergings in trajectory streams is a challenging task. First, it is hard to discover all moving objects attending a gathering event in the same way as many state-of-the-art trajectory clustering methods since they do not stay close together most of the time. Second, to achieve early detection, converging algorithms need to detect gathering events in an incremental manner. Third, identifying cluster containment relations between any consecutive timestamps may face huge search space and hence incur high cost. Fourth, the system has to ensure the effectiveness of mining converging patterns from trajectory streams.

To tackle these issues, the GEDetector employs a general framework for effective and efficient discovery of convergings. Specially, To boost the performance of CCJ, which is a fundamental part of mining converging patterns, we develop a signature quad-tree based index, called SQTI, to organize clusters hierarchically and spatially, and correspondingly propose an SQTI based CCJ (SQTCCJ) approximate algorithm, which enables us to rapidly filter unqualified candidates and efficiently identify matches by considering cluster containment relationship and spatial proximity simultaneously. In addition, to facilitate evaluating set containment, we approximate the sets of moving objects by means of a signature technique, and use a bloom filter method [8] to generate signatures in this work.

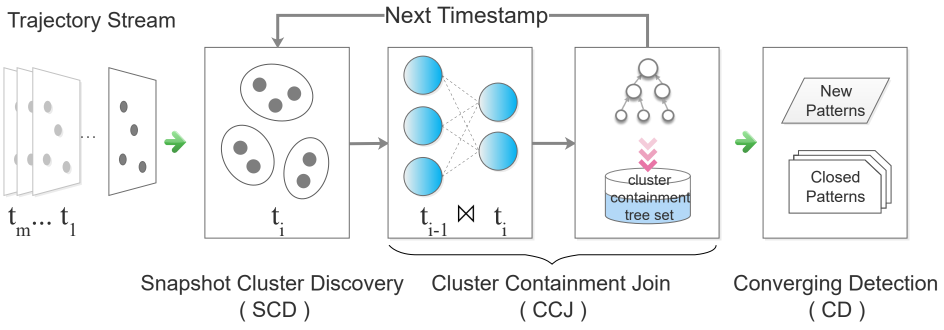
The SQTCCJ consists of two phases: (1) SQTI construction; and (2) SQTI probing and verification. Firstly, we construct an SQTI structure to organize all candidate clusters. Then, SQTCCJ performs a cluster query on SQTI for each query cluster to obtain all cluster containment matches.

SQTI Construction. An SQTI is constructed in the following manner. We start with a set of candidate clusters, and insert representative points of these clusters into a spatial region according to their positions. Then we partition the whole spatial region by recursively subdividing it into four square quadrant cells (i.e., NE, NW, SE, and SW), until the number of points in it meets a certain threshold, ρ. Otherwise, the quadrant cell continues splitting into four small ones. The tree structure of SQTI follows the above spatial decomposition. Although the SQTI is organized in the same way as the traditional quad-tree, the difference is that each node is assigned a signature consisting of m bits, which represents a set of moving objects of all clusters in the corresponding cell. In the next phase, we will utilize signature comparisons to support membership query.

Example 1. *We use the data in Fig. 3(a)(b) to illustrate the construction of SQTI. Based on these clusters, we build an SQTI for searching clusters, and its nodes are organized as Fig. 3(c)(d) shows.*

SQTI Probing and Verification. The search process based on SQTI works in a top-down manner instead of in a recursive manner. And its goal is to find the nearest quadrant cell to the query point. Specifically, when a query cluster q comes, the search starts at the root node and utilize the query signature and its coordinates to probe the index structure. When it visits a internal node p with signature $p.sig$, we need to check it to see if $q.sig \vee p.sig = p.sig$. If yes, it immediately performs a four-way comparison operation at the node, and then chooses the subtree where the centroid of its corresponding MBR is nearest to the query point. Otherwise, it visits the sibling nodes. When it reaches a leaf node, we need to verify all clusters to check if there exists a super-cluster in the leaf node. If yes, we can obtain a cluster containment match; otherwise, we get a mismatch.

Example 2. *Continuing with Example 1 , we illustrate how to perform the query process based on SQTI. For a given query*

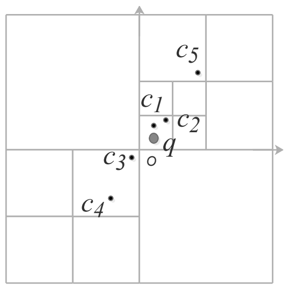


Trajectory Stream
Next Timestamp
$t_m \ldots t_1$
t_i
$t_{i-1} \bowtie t_i$
cluster containment tree set
New Patterns
Closed Patterns
Snapshot Cluster Discovery (SCD)
Cluster Containment Join (CCJ)
Converging Detection (CD)

Figure 2: Framework of GEDetector

candidate cluster	signature
c_1: $\{o_1, o_2, o_3\}$	01110000
c_2: $\{o_4, o_5, o_6\}$	00001110
c_3: $\{o_7, o_8\}$	10000001
c_4: $\{o_9, o_{10}, o_{11}\}$	01110000
c_5: $\{o_{12}, o_{13}\}$	00001100

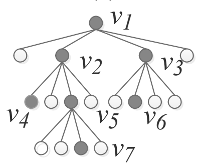
(a)



(b)

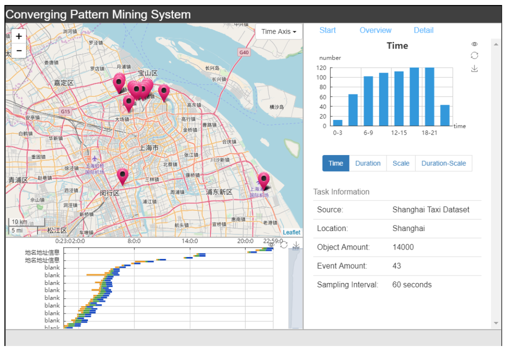
node in SQTI	signature
v_1: $\{c_1, c_2, c_3, c_4, c_5\}$	11111111
v_2: $\{c_1, c_2, c_5\}$	01111110
v_3: $\{c_3, c_4\}$	11110001
v_4: $\{c_5\}$	00001100
v_5: $\{c_1, c_2\}$	01111110
v_6: $\{c_3, c_4\}$	11110001
v_7: $\{c_1, c_2\}$	01111110
other nodes	00000000

(c)



(d)

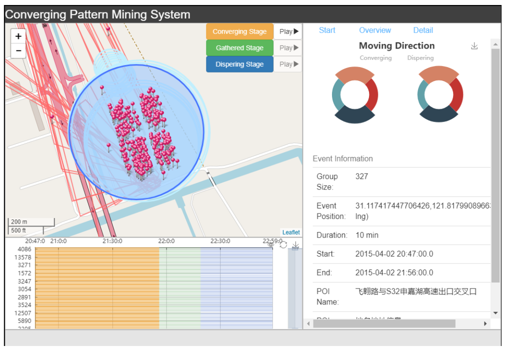
Figure 3: Illustration of SQTI construction



Task Information

Source:	Shanghai Taxi Dataset
Location:	Shanghai
Object Amount:	14000
Event Amount:	43
Sampling Interval:	60 seconds

Figure 4: Screenshot of main interface



Event Information

Group Size:	327
Event Position:	31.117417447706426,121.8179908966 lng)
Duration:	10 min
Start:	2015-04-02 20:47:00.0
End:	2015-04-02 21:56:00.0
POI Name:	飞翔路与S32申嘉湖高速出口交叉口

Figure 5: Visualization of a taxi gathering event at Shanghai Pudong international airport

$q = \{o_1, o_2\}$ with the signature 01100000*, the search starts at the root node v_1. Then it need to check the nodes v_2 and v_3 to see if the query signature is contained. Unfortunately, both v_2 and v_3 satisfy the above condition. In this case, the search algorithm in the traditional tree will suffer from a backtracking problem incurred by a recursive paradigm. But, the search in SQTCCJ can avoid this since it only takes the nearest quadrant cell to the query position. As a consequence, the search chooses the sub-tree rooted at the node v_2. It reaches the leaf node v_7 along the path $v_1v_2v_5v_7$ after signature comparisons. At the leaf node, we still need to verify all clusters in it because of false positive. Finally, we can obtain a match $q \subseteq_c c_1$.*

4 DEMONSTRATION

4.1 Demo Interface

In the demo of GEDetector, we provide a monitoring map interface for users to visualize and monitor gathering events dynamically. A screenshot of the demo is shown in Fig. 4. This demo interface allows users to investigate gathering events from various views, including spatial view, temporal view, and detail view. The map on the upper left pane of the GUI visualizes the discovered gathering events using map markers, which represent the location points of all gathering events. The bar chart on the bottom left pane of the GUI represents the basic characteristics of gathering events, such as scale, start time, durability, and point of interest (POI) type. When the user clicks on any map marker, the interface switches to the display mode of the corresponding gathering event. The user can obtain more detailed information of the gathering event, including the routes of all participators, direction statistics of all routes (represented by a wind rose diagram), the event timeline, and other properties. As shown in Fig. 5 and 6, our system has detected a typical gathering event at Shanghai Pudong international airport, where a large number of taxis converge together for waiting guests.

In summary, the GEDetection system can allow users to interactively monitor and visualize gathering behaviours in various ways, and also help users gain insights of the identified gathering events from both spatial and temporal aspects.

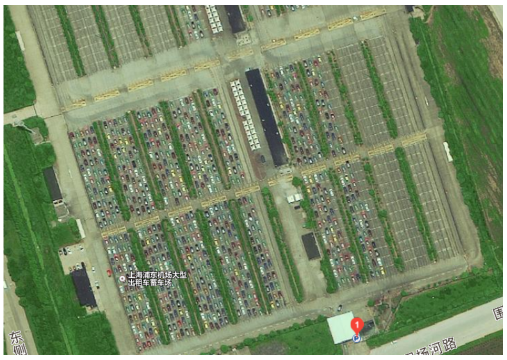


Figure 6: Satpic of a cabstand at Shanghai Pudong international airport

5 PERFORMANCE OF GEDETECTOR

To evaluate the performance of our system, we conduct all experiments based on a real trajectory dataset, namely Shanghai taxi cab traces (BigTaxi), which is sourced from the one generated by 13,000 taxis of Shanghai in a period of 30 days (from April 1st to April 30th in 2015). In this section, we investigate the efficiency of the SQTCCJ algorithm, which is the essential part of our system.

The SQTCCJ algorithm may generate approximate results due to missing some cluster containment matches, only when a result cluster does not share the same cell with the query. Therefore, we use an accuracy rate metric, which is the fraction of matches obtained by SQTCCJ over the total amount of those of NLCCJ, to evaluate the result of SQTCCJ algorithm. In Fig. 7(a), we show accuracy rates of SQTCCJ. We can find that SQTCCJ can find almost all candidate clusters when the parameter ρ is greater than 20.

Next, in Fig. 7(b)(c), we show another nice property of SQTCCJ, namely insensitivity to key parameters including the signature length m and the number threshold ρ. In addition, we note that we can tackle the problem of CCJ using set containment join methods if we treat a cluster as a set. Thus, we compare our algorithm with TT-Join [9], which is the state-of-the-art method evaluating set containment join. As we can see from Fig. 7(d), SQTCCJ significantly outperforms TT-Join especially as the dataset size increases.

6 ACKNOWLEDGEMENTS

This study was supported by the National Natural Science Foundation of China under grant numbers 41471371, 41571379, and the PAPD program.

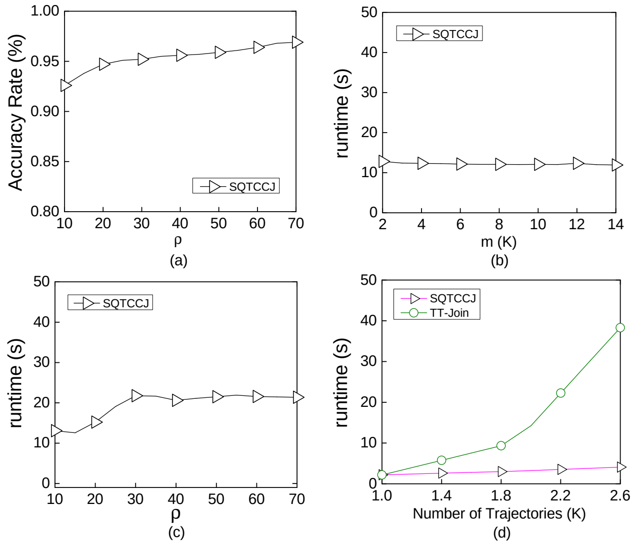


Figure 7: Performance of SQTCCJ

REFERENCES

[1] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups of trackable moving point objects. In *Geographic Information Science, Second International Conference, GIScience 2002, Boulder, CO, USA, September 25-28, 2002, Proceedings*, pages 132–144, 2002.

[2] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of Convoys in Trajectory Databases. *Proc. VLDB Endow.*, 1(1):1068–1080, August 2008.

[3] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining Relaxed Temporal Moving Object Clusters. *Proc. VLDB Endow.*, 3(1-2):723–734, September 2010.

[4] L. A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C. C. Hung, and W. C. Peng. On Discovery of Traveling Companions from Streaming Trajectories. In *2012 IEEE 28th International Conference on Data Engineering (ICDE)*, pages 186–197, April 2012.

[5] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 242–253, April 2013.

[6] Qi Fan, Dongxiang Zhang, Huayu Wu, and Kian-Lee Tan. A General and Parallel Platform for Mining Co-movement Patterns over Large-scale Trajectories. *Proc. VLDB Endow.*, 10(4):313–324, November 2016.

[7] Yu Zheng. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):1–41, May 2015.

[8] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[9] Jianye Yang, Wenjie Zhang, Shiyu Yang, Ying Zhang, and Xuemin Lin. Tt-join: Efficient set containment join. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 509–520, 2017.