Gathering Pattern Mining Method Based on Trajectory Data Stream

Ying Xia[1(✉)], Lian Diao[1], Xu Zhang[1], and Hae-young Bae[2]

[1] School of Computer Science and Technology,
Chongqing University of Posts and Telecommunications, Chongqing, China
{xiaying, zhangx}@cqupt.edu.cn,
S160201017@stu.cqupt.edu.cn
[2] Department of Computer Engineering, Inha University, Incheon, South Korea
hybae@inha.ac.kr

Abstract. Moving object gathering pattern refers to a group of incident or case that are involved large congregation of moving objects. Mining the moving object gathering pattern in massive and dynamic trajectory data streams can timely discover the anomalies in the group moving model. This paper proposes a moving object gathering pattern mining method based on trajectory data stream, which consists of two stages: clustering and crowed mining. In the clustering stage, the MR-GDBSCAN clustering algorithm is proposed. It uses the grid to index moving objects and uses the grid as a clustering object and determines the center of each cluster. In the crowed mining phase, the sliding time window is used for incremental crowed mining, and the cluster center is used to calculate the distance between different clusters, thereby improving the crowed detection efficiency. Experiments show that the proposed moving object gathering pattern mining method has good efficiency and stability.

Keywords: Gathering pattern · Trajectory data streams · Clustering · Crowed · Sliding time window

1 Introduction

In recent years, with the development of mobile positioning, Internet, cloud computing, big data technology and the popularity of smart terminals. Location-based information services are increasingly enriched, and moving object trajectory data is continuously concentrated in many information service platforms, such as website check-in data, mobile signaling data, vehicle-mounted GPS data, RFID data, etc. The moving object gathering pattern mining of massive and dynamic trajectory data can timely find some moving object groups that are close to each other and move together [1], and provide services for traffic dispatching, public safety and other fields.

Many researchers have studied the moving object gathering pattern, such as flock [2, 3], moving cluster [4], convoy [5, 6], swarm [7], companion [8]. As the research progresses, its conditional constraints are more in line with human behavior patterns. Zheng et al. [1] proposed a gathering pattern, which consists of clusters of at least n time slices, and requires at least m_p participator in each cluster. This method firstly establishes

J. Li et al. (Eds.): SPNCE 2019, LNICST 284, pp. 666–676, 2019.
https://doi.org/10.1007/978-3-030-21373-2_56

a grid index on the trajectory data, then clusters the moving objects on each time slice, and proposes a closed crowed concept to detect the moving object gathering pattern. The concept of participator proposed makes the pattern more suitable for the behavior patterns of moving objects in real life. Zhang et al. [9] proposed a gathering pattern mining method based on spatio-temporal graph. It finds the largest complete subgraph that satisfies the spatiotemporal constraints in the spatio-temporal graph composed of clusters, and then mines the gathering pattern at a given time or position. Zhang et al. [10] considered that the gathering pattern is defined based on the "co-occurrence" pattern. It has a "parking problem", and the mining results are mixed with a large number of non-moving aggregated objects. To solve this problem, a converging pattern based on group motion process modeling is proposed. Yang et al. [11] used a quadtree to index data in the DBSCAN clustering algorithm to improve the mining efficiency of the converging pattern.

It can be seen that the moving object gathering pattern mining method is continuously improved with the change of data characteristics such as static data, dynamic data, big data, and streaming big data. Therefore, how to conduct efficient gathering mode mining in big data and streaming environment is the current research hotspot. This paper focuses on trajectory big data with streaming features, and relies on the Spark Streaming [12] framework to efficiently mine the gathering pattern of moving objects from the trajectory data stream. In the streaming environment, data is continuously received and gathering results are continuously output. If only the closed-crowed [1] is found, it often requires a large delay to obtain the gathering result, and it is not well adapted to the service requirements with high timeliness such as road condition analysis and route planning.

The main contributions of this article are: (1) A grid-based clustering algorithm is proposed. The algorithm first maps each moving object into a unique grid, and records the number of moving objects in the grid as weight, and then clusters the weighted grid to improve the efficiency of clustering. (2) An incremental open crowed detection algorithm based on sliding time window is proposed. First, set the sliding time window width and the window sliding distance, then cluster each time window data in the sliding time window, and then mine the crowed in the sliding time window. With the newly arrived time window data, the crowed detection result of the last sliding time window is multiplexed to update the crowed detection result of the current sliding time window.

The remainder of the paper is organized as follows. The relevant definitions of the moving object gathering pattern are given in Sect. 2. A gathering pattern mining method based on trajectory data stream is described in detail in Sect. 3. We evaluate the efficiency and scalability of the method in Sect. 4, and summarize the full text in Sect. 5.

2 Definition

For the convenience of description, the related concepts and symbols are first defined.

2.1 Snapshot Trajectory

Given the moving object database O_{DB} and the time domain of database T_{DB}, the snapshot trajectory $Sli_i = \{(o_i, t_i) | o_i \in O_{DB}, t_i \in T_{DB}\}$, Sli_i is the snapshot trajectory of the moving object o_i in the t_i time slice. It belongs to a subset of the snapshot trajectory set Sli.

2.2 Core Grid

Given the distance threshold σ, and the cluster density threshold m_c, the $grid_{(x,y)}$ is a core grid when any of the following conditions are satisfied:

1. The number of moving objects in the grid $grid_{(x,y)}$ is more than m_c.
2. The grid in the neighborhood of the grid $grid_{(x,y)}$ contains more moving objects than m_c.

2.3 Snapshot Cluster

Let the set $c_i = \{o_1, o_2, \ldots, o_\mathrm{i}\}$ be a set of moving objects of a time slice i $(1 \leq \mathrm{i} \leq \mathrm{k})$. For a given cluster density threshold m_c, and a distance threshold σ. c_i is a snapshot cluster when the following conditions are satisfied:

1. $\mathrm{size}(c_i) \geq m_c$.
2. The distance between the members of c_i in the grid g_k and g_p is not greater than σ, $\mathrm{Dist}(g_\mathrm{k}, g_\mathrm{p}) \leq \sigma$.

2.4 Crowed

Given the moving object database O_{DB}, the cluster density threshold m_c, the distance threshold σ, and the lifetime threshold k_c, a crowed C_r is a sequence of snapshot clusters at consecutive timestamps, i.e., $C_r = (c_{t_a}, c_{t_{a+1}}, \ldots, c_{t_b})$. The following conditions need to be satisfy [1]:

3. The lifetime of C_r is defined as $C_r.\tau$, is not less than k_c, i.e., $C_r.\tau = \mathrm{b} - \mathrm{a} + 1 \geq k_c$.
4. There are at least m_c objects at any time, i.e., $\forall \mathrm{a} \leq \mathrm{i} \leq \mathrm{b}, |c_{i_i}| \geq m_c$.
5. The distance between two consecutive snapshot clusters is not greater than 2σ, i.e., $d_H(c_{t_i}, c_{t_{i+1}}) \leq 2\sigma, \forall \mathrm{a} \leq \mathrm{i} \leq \mathrm{b} - 1$.

2.5 Participator

In crowed C_r, a moving object can be referred to as a participator, which appears at least k_p times in the crowed [1].

2.6 Gathering

A crowed is a gathering, which has at least m_p participator in each snapshot cluster [1], i.e., $\forall c_t \in C_r, |\{o|o(t) \in c_t, o \in \mathrm{Par}(C_r)\}| \geq m_p$.

Figure 1 is an example of the movement trajectory of seven moving objects in seven time slices. By definition, o_1, o_2, o_3, o_4 form a gathering, and o_5, o_6, o_7 form an another gathering. The symbols used in this paper are listed in Table 1.

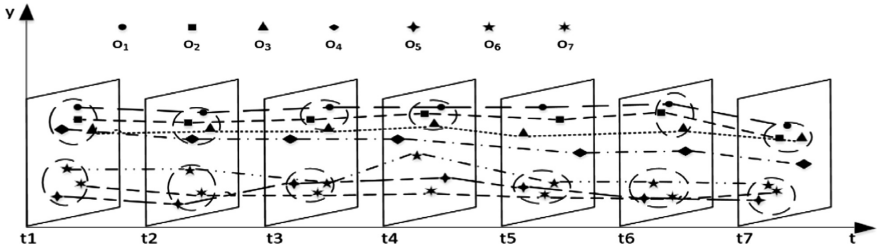

Fig. 1. Moving object gathering pattern.

Table 1. Table of notations.

Notation	Definition	Notation	Definition
O_{DB}	Moving object database	$d_H(c_i, c_j)$	Distance between clusters c_i, c_j
T_{DB}	Time domain of database	k_c	Lifetime threshold of a crowed
t_i	A time point in T_{DB}	m_c	Cluster density threshold
o_i	A moving object	$\mathrm{Par}(C_r)$	Participator set of a crowed C_r
Sli_i	A snapshot trajectory at time t_i	k_p	Lifetime threshold of a participator
c_i	A snapshot cluster at time t_i	m_p	Support threshold of a gathering
C_r	A crowed	σ	Distance threshold
$C_r.\tau$	The lifetime of a crowed		

3 Gathering Pattern Mining Method Based on Trajectory Data Stream

In the streaming environment, the trajectory data is continuously received, we continuously cluster in each time window, and perform crowed detection in the sliding time window. If the crowed is detected in turn for each time window in each sliding time window, the efficiency is often unsatisfactory. Therefore, it is considered to use the incremental detection method to perform crowed mining on the real-time arrival time window. The structural framework of the method described in this paper is shown in Fig. 2, which mainly includes two parts: moving object clustering and crowed detection.

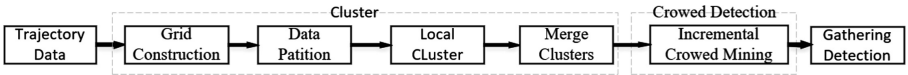

Fig. 2. Spark Streaming-based trajectory data stream gathering pattern mining method framework.

3.1 Grid-Based Clustering Algorithm

Clustering is the key to crowed mining. In order to improve the clustering efficiency, a grid-based clustering algorithm (Grid-based MR-DBSCAN, MR-GDBSCAN) is proposed based on MR-DBSCAN [13], which clusters moving objects in each time window. Given the data set O_{DB}, the algorithm is divided into four steps:

The first step is to generate a grid structure $grid_{(x,y)}$, where x represents the abscissa and y represents the ordinate. The grid structure determines that the maximum number of grids does not exceed $x * y$. For each moving object, it is assigned to the corresponding grid according to its latitude and longitude coordinates.

The second step is data partitioning. In the distributed environment, data partitioning may make the results of clustering inaccurate, especially those that are close to the partition boundaries. For example, the objects o_1, o_2, o_3 belong to c_i, but if the data is partitioned, o_1, o_2 are partitioned in the same partition, but o_3 is in another partition. It causes the clustering results in the partition to be inconsistent with the clustering results on the entire snapshot cluster, but it is less efficient if all the data is clustered on one partition. Therefore, the partitioning method in the MR-DBSCAN algorithm is used for processing.

The third step is local clustering. Using grids as clustering objects instead of moving objects can greatly improve clustering efficiency.

The fourth step is to merge clusters. Using the MR-DBSCAN algorithm, the clusters in each partition are combined to obtain the global clustering result.

Grid Construction. Given a data set O_{DB}, each point has its own grid. It is calculated as follows:

$$\left(\mathrm{x} = \left\lceil \frac{x - x_{min}}{\partial} \right\rceil, \mathrm{y} = \left\lceil \frac{y - y_{min}}{\partial} \right\rceil\right) \tag{1}$$

where ∂ is the length of the grid, the value is half of σ, x_{min} and y_{min} are the minimum longitude and minimum latitude of O_{DB}. We define n_{xy} to represent the weight of $grid_{(x,y)}$, which is the number of moving objects contained in the grid. Once all the data has been mapped to the corresponding grid, the weight of the grid can be determined. Then, the average value of the latitude and longitude is calculated by the coordinates of all the moving objects of the grid as new grid coordinates, so that the grid coordinates change with the distribution of the objects in the grid. The final data format looks like this: $\langle \mathrm{newgrid}, \mathrm{time}, \mathrm{oldgrid}, n_{xy} \rangle$, where newgrid represents the grid id (including x coordinate and y coordinate), which is the average calculated grid coordinate, time indicates the time slice number, oldgrid indicates the old grid id. Figure 3 is an example of a grid construction where the number is the weight of the grid.

Local Clustering. The DBSCAN clustering algorithm can identify whether the subset is a cluster based on the subset density of a data set, and can identify cluster clusters of arbitrary shapes, but its time complexity $\mathrm{O}(n^2)$ is unacceptable when clustering

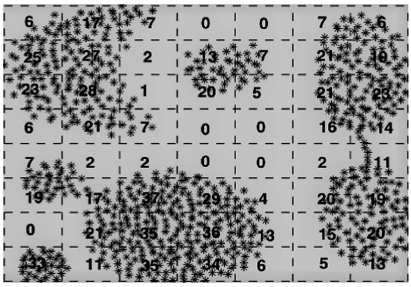

Fig. 3. Grid construction

massive amounts of data. In the process of local clustering, using the grid as the clustering object instead of using the point as the clustering object makes the clustering efficiency greatly improved. Figure 4 shows an improved local clustering algorithm.

The algorithm starts from any unvisited grid p in the dataset and traverses in order. Second find all the grids in the neighborhood of the grid p as its neighbor grid, and then sum the weight of its neighbor grids. If the sum is smaller than the parameter Minpts, the grid p is marked as a noise grid, and then turn to next grid. Otherwise p is a core grid, build cluster c_p based on grid p, and then perform the same detection and expansion on its neighbor grid.

Algorithm 1：Local clustering	
Input:	Dataset *D*, Radius *Eps*, Density threshold *Minpts*
Output:	Cluster *C*
1.	$C \leftarrow \emptyset$
2.	**for** each unvisited grid p in D **do**
3.	Mark p as visited
4.	*NG* = getNeighbourGrid(p, Eps) // find the neighbor grid
5.	*pointInNG* = Sum(N) // count the weight of the neighbor grid
6.	**if** *pointInN* < *MinPts* **then**
7.	Mark p as Noise
8.	**else**
9.	*C* = next cluster
10.	ExpandCluster(p, NG, C, Eps, MinPts)
11.	**Return** C

Fig. 4. Local clustering algorithm

The time complexity of the algorithm is $O(\sum_{i=1}^{partition}(|S_i| * \text{getNeighbor-Grid}()))$ where partition is the number of partitions, $|S_i|$ is the number of grids in each partition,

and GetNeighborGrid() is the time to get its neighbor grids. Therefore, the time complexity of the algorithm depends on the number of partitions, the number of grids in the partition, and the time it takes to find its neighbors.

3.2 Incremental Crowed Mining Algorithm Based on Sliding Time Window

In the detection process of the crowed, the concept of sliding time window is used, and the crowed is mined in the sliding time window. When the new time window arrives, it will reuse the crowed mining result of the last sliding time window to update the crowed result of the current time window, which saves space and reduces the amount of calculation. Figure 5 is a sliding time window model, which uses three times windows as the calculation window of the sliding time window, and slides one time window distance. Figures 6 and 7 briefly describe the *func* function and the *invFunc* function for incremental mining crowed that rely on the *reduceByKeyAnd Window (func, invFunc, windowLength, interval)* functions of the Spark Streaming framework, where *windowLength* represents the length of the sliding time window, *interval* represents the sliding distance.

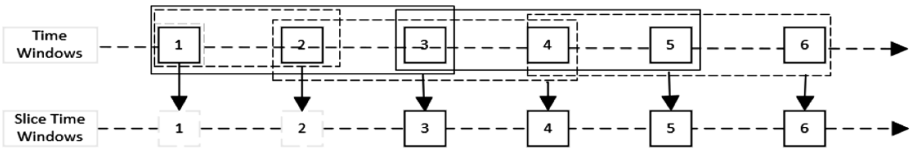

Fig. 5. Sliding time window model

Relying on the *reduceByKeyAndWindow* function, it will call the *func* function to perform crowed mining for each time window. When the sliding time window size is equal to its set value, the mining result of the current sliding time window will be output. When the fourth time window comes, the *invFunc* function is called to process the candidate crowed result of the last sliding time window, which will get a new crowed candidate set. Then, this new crowed candidate set will be combined with the cluster in the current time window to expand into a new crowed.

In daily life, a large gathering, parade and other activities, the flow of people is often very slow, gradual, rather than mutated [1], the shape and size of the population changes relatively slowly. Therefore, the cluster center can often represent the position of the cluster and can be used to calculate the distance between the cluster and the cluster, which can improve the calculation efficiency.

Algorithm 2：*func* (*CC*, *CW*)

Input: Crowed candidate set for the current sliding time window CC
Cluster set of the next time window CW
Output: Crowed set within the current sliding time window CS
1. $CS \leftarrow \emptyset$
2. $addClaster \leftarrow \emptyset$
3. **for** each crowed candidate *cc* in *CC* **do**
4. $Cr' \leftarrow$ FindCluster(*cc, CW*) //find the set if clusters that are within 2σ distance to cc.
5. $addClaster \leftarrow$ addClaster $\cup$ Cr'
6. **if** $Cr' = \emptyset$ **then**
7. $CS \leftarrow CS \cup cc$
8. **else**
9. **for** each *c* in Cr' **do**
10. $cc \leftarrow cc \cup c$
11. $CS \leftarrow CS \cup cc$
12. $CS \leftarrow CS \cup$ notAddedCluster(*addCluster, CW*) //the clusters cann-ot be appended to any current crowed camdidate will become new crowed camdidate.
13. **return** CS

Fig. 6. Crowed detection algorithm

Algorithm 3: *invFunc* (*CC*, *CW*)

Input: Crowed mining results from the last sliding time window *CC*
The cluster in Time window that is discarded in the current sliding time window *CW*
Output: Duplicate crowed set *DC*
1. $DC \leftarrow \emptyset$
2. **for** each window *w* in *CW* **do**
3. **for** each cluster c in w **do**
4. $DC \leftarrow$ updateCrowed(*c*, *CC*) //find the cluster in the Crowed and delete it in the Crowed.
5. **return** *DC*

Fig. 7. Update crowed algorithm

4 Experiment

The experiment used data provided by Datacastle's Traffic Line Time Prediction Competition [1], which contains more than 1.4 billion GPS records for more than 14,000 taxis in Chengdu. The data is recorded from August 03, 2014 to August 30, 2014, and the upload interval is one minute. We attach a time attribute to the car id to

bring all the data to one day. In this experiment, we generate 3 sets of data, D1 has up to 250,000 moving objects, D2 has up to 600,000 moving objects, and D3 has up to 1 million moving objects. As shown in Fig. 8, the amount of data processed in each time window in each data set simulates a business activity, with a flow of people from more to less. In the data stream, we send data to Kafka at intervals of 1 min, and then read the data through Spark Streaming for processing. The experimental cluster consists of four hosts. Its CPU is 4-core Intel(R) Xeon(R)E2430@2.2 GHz, 8G memory, running Linux operating system, building Hadoop 2.60 and Spark1.60 distributed cluster.

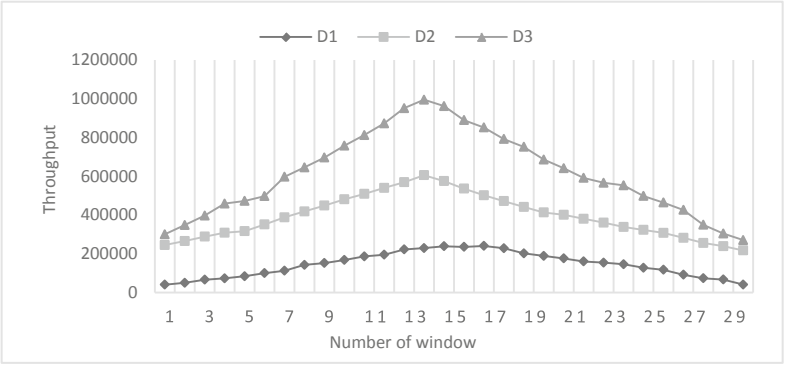

Fig. 8. The amount of data processed in each time window

4.1 Efficiency Evaluate

Given the data set D1, the time database $|O_{DB}|$ contains 30 time slices, First set the cluster density threshold m_c = 10, sliding time window = 6 min, window sliding distance = 1 m, time threshold = 4 min, distance threshold σ take GPS coordinate interval = 0.002, the distance is about 200 m, crowed clustering distance threshold μ = 2σ. Figure 9 shows the gathering pattern mining processing time in which the sliding time window has six-time windows and the sliding distance is one time window. The processing time is obtained by observing the Spark UI. As can be seen from the figure, as the amount of data in each time window increases, the change of the mining time of the gathering pattern based on the MR-DBSCAN algorithm and the Crowed-TAD algorithm (abbreviated as Crowed-TAD) is consistent with the change of the data amount. It has good mining efficiency when the amount of data is small. However, with the change of data volume, the processing time of the gathering pattern detection method (abbreviated as Proposed) based on MR-GDBSCAN algorithm and incremental open crowed detection algorithm based on sliding time window remains stable, and the gathering pattern mining is more efficient.

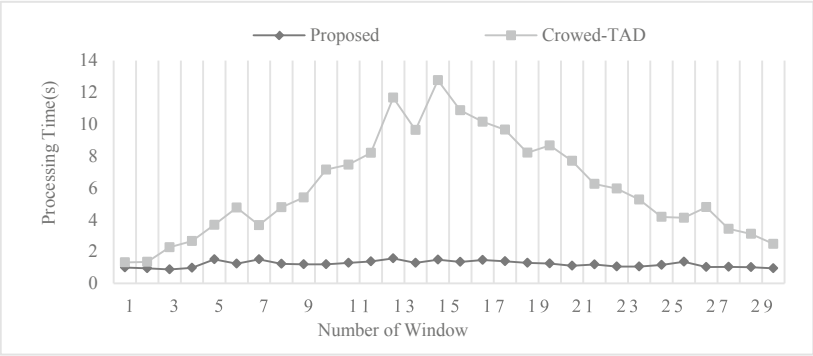

Fig. 9. Gathering pattern mining processing time

4.2 Scalability Evaluate

On the datasets with different data volumes, the Proposed method is evaluated for scalability, and its parameters use the parameter settings in the efficiency evaluation. As can be seen from Fig. 10, as the amount of data in the data set continues to increase, although the detection time increases, the gathering pattern mining efficiency remains within an acceptable range.

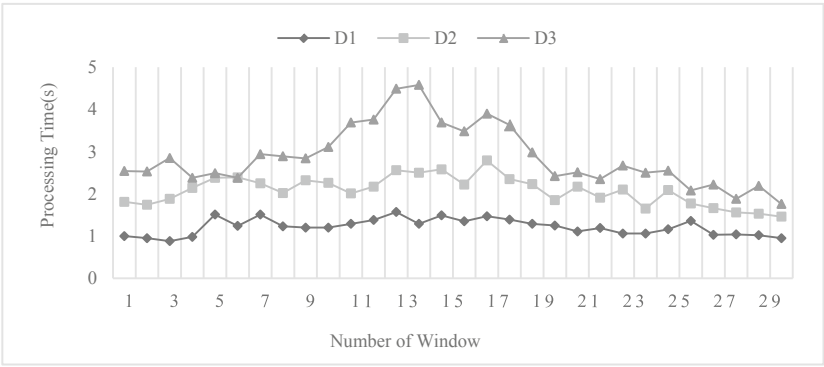

Fig. 10. Processing Time evaluate of different data sets

5 Conclusion

Moving object gathering pattern is a research hotspot in mobile object pattern mining. In order to adapt to the efficient gathering pattern mining requirements in big data and streaming environment, this paper proposes a moving object gathering pattern mining method based on trajectory data stream. The method first optimizes the traditional DBSCAN clustering algorithm, which uses grids to cluster and make clustering more

efficient. Then incremental crowed detection is performed on the sliding time window to improve the efficiency of the gathering pattern retrieval. Experiments show that this method has better advantages in mining efficiency and scalability.

References

1. Zheng, K., Zheng, Y., Yuan, N.J., Shang, S., Zhou, A.X.: Online discovery of gathering patterns over trajectories. IEEE Trans. Knowl. Data Eng. **26**(8), 1974–1988 (2013)
2. Wang, Y., Lim, E.-P., Hwang, S.-Y.: On mining group patterns of mobile users. In: Mařík, V., Retschitzegger, W., Štěpánková, O. (eds.) DEXA 2003. LNCS, vol. 2736, pp. 287–296. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45227-0_29
3. Wang, Y., Lim, E.-P., Hwang, S.-Y.: Efficient group pattern mining using data summarization. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 895–907. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24571-1_78
4. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 364–381. Springer, Heidelberg (2005). https://doi.org/10.1007/11535331_21
5. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of convoys in trajectory databases. Comput. Sci. **1**(1), 1068–1080 (2009)
6. Aung, H.H., Tan, K.-L.: Discovery of evolving convoys. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 196–213. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13818-8_16
7. Li, Z., Ding, B., Rol, J.H.: Swarm: mining relaxed temporal moving object clusters. Proc. VLDB Endow. **3**(1–2), 723–734 (2010)
8. Tang, L.A., et al.: On discovery of traveling companions from streaming trajectories. In: 2012 IEEE 28th International Conference on Data Engineering. IEEE Computer Society (2012)
9. Zhang, J., Li, J., Liu, Z., Yuan, Q., Yang, F.: Moving objects gathering patterns retrieving based on spatio-temporal graph. Int. J. Web Serv. Res. **13**(3), 88–107 (2016)
10. Yu, Y., Genlin, J., Bin, Z., Xiaoting, H.: A new algorithm for mining gathering pattern from spatio-temporal trajectories. J. Nanjing Univ. **54**(1), 97–106 (2018)
11. Yifan, Z., Bin, Z., Hongyan, S., Chao, T., Genji, J.: Algorithm for mining converging patterns of moving objects from spatiotemporal trajectories. J. Data Acquis. Process. **33**(3), 487–495 (2018)
12. SparkStreaming homepage. http://spark.apache.org/streaming/. Accessed 30 Nov 2018
13. He, Y., et al.: MR-DBSCAN: an efficient parallel density-based clustering algorithm using MapReduce. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems, pp. 473–480 (2011)