Discovery of Gathering Patterns of Moving Objects

Ravi Raj Gupta[1], T. Ramakrishnudu[2]
[1, 2]Department of Computer Science and Engineering
National Institute of Technology
Warangal, A.P.-506 004. India

ABSTACT: *The increasing pervasiveness of location-acquisition technologies has embedded collection of huge amount of trajectories for almost any kind of moving objects. Discovering useful patterns from their movement behaviors can convey valuable knowledge to a variety of critical applications. In this light we propose a concept, called gathering, which is a trajectory pattern modeling various group incidents such as celebrations, parades, protests, traffic jams and so on. In this work, we first develop a set of techniques to tackle the challenge of efficient discovery of gathering patterns on archived trajectory dataset. For finding gathering firstly we have to find snapshot cluster, crowd and then super crowd. After getting super crowd, gathering can be identified. Here we proposed an efficient algorithm for finding super crowd if cluster database is given and algorithm for identify gathering after identifying the super crowd. Afterwards, since trajectory databases are inherently dynamic in many real-world scenarios such as traffic monitoring, fleet management and battlefield surveillance, we further propose discovery solution by applying a series of optimization schemes to handle the incremental data.*

Keywords: Trajectory database, Pattern mining, Cluster, Crowd, Gathering Patterns

Received: 24 February 2015, Revised 20 March 2015, Accepted 28 March 2015

1. Introduction

The increasing availability of location-acquisition technologies including telemetry attached on wildlife, GPS set on cars, WLAN networks, and mobile phones carried by people have enabled tracking almost any kind of moving objects, which results in huge volumes of spatio-temporal data in the form of trajectories. Such data provides the opportunity of discovering usable knowledge about movement behaviour, which fosters ranges of novel applications and services [1]. For this reason, it has received great attention to perform data analysis on trajectories. In this paper, we move towards this direction and address one particular challenge to do with discovering the so-called gathering patterns from trajectories in an efficient manner.

Informally, a gathering represents a group event or incident that involves congregation of objects (e.g., vehicles, people, animals). A gathering is expected to imply something unusual or significant happening. As such, the discovery of gatherings over trajectories can help in sensing, monitoring and predicating non-trivial group incidents in everyday life. However, discovering the gatherings from trajectories is not an easy task, where challenges are two-fold. First, how to define the concept of gathering

appropriately such that it intuitively captures the properties of the above mentioned events, while being rigid from algorithmic aspect in the mean time. Second, how to develop a solution that can discover gatherings from large scale trajectories efficiently, and more importantly, handle new data arrivals in an incremental manner.

Let $O_{DB} = \{o_1, o_2, o_3, \ldots\ldots, o_n\}$ be the set of all moving objects in the database and $T_{DB} = \{t_1, t_2, t_3, \ldots\ldots, t_m\}$ be the time domain, where each t_i is a time point. The trajectory of moving object during time interval $[t_1, t_n]$ is represented by $o = \{(p_1, t_1), (p_2, t_2), (p_3, t_3), \ldots\ldots, (p_n, t_n)\}$. Where p_i represents location of object at time t_i. Here the user will input threshold distance d and an integer value m, to find the snapshot cluster c_t at any timestamp t. Then user will input a threshold value k_c to find crowd. A crowd C_r is a set of sequence of snapshot cluster at consecutive timestamp i.e. $C_r = \{c_{ta}, c_{ta+1}, c_{ta+2}, \ldots\ldots c_{tb-1}, c_{tb}\}$. A crowd C_r is called a gathering if there exist at least m_p participator in each snapshot cluster. A gathering is said to be closed if there is no super-crowd of *Cr* that is also a gathering.

2. Methods

2.1 Discovering Snapshot Cluster

Object database and timestamp database are given. So Now we adopt the notion of density-based clustering [2][3] to define the snapshot cluster[4][5], for this two threshold values will be needed i.e. threshold distance d and an integer value m. we will apply the concept of d-neighbourhood, direct density reachable and direct density reachable. A snapshot cluster is a group of objects with arbitrary shape and size, which are density-connected to each other at a given timestamp. This way we will generate cluster database C_{DB}.

If p be a point then d-neighbourhood of point p is defined as $N_d = \{q \in S \mid D(q, p) \leq d\}$. Here $D(\,,\,)$ represents Euclidean distance between two points p and q. If Euclidean distance between two points p and q are less then threshold value d then p and q are called directly density reachable with respect to threshold distance d and threshold value m. Same way p and q are called density reachable with respect to d and m if there exist some point objects between p and q like $p_1, p_2, p_3, p_4, \ldots. p_n$ such that each consecutive pair between p and q including p and q are direct density reachable.

2.2 Discovering Crowd

A crowd C_r is a set of sequence of snapshot cluster at consecutive timestamp i.e. $C_r = \{c_{ta}, c_{ta+1}, c_{ta+2}, \ldots\ldots c_{tb-1}, c_{tb}\}$. Since a snapshot cluster is essentially a set of points, we adopt the Hausdorff distance [6][7] to measure how far two clusters are from each other For crowd we need lifetime threshold value k_c, threshold distance δ. The lifetime of a crowd is denoted by C_r and it should not be less than threshold lifetime.

1) Time $(b - a + 1)$ should be greater than or equal to k_c.

2) Distance between any consecutive pair of snapshot should not be greater than δ.

$$Dist(c_{ti}, c_{ti+1}) \leq \delta, \forall a \leq i \leq b - 1$$

One thing should be noted that no super set of C_r should be a crowd. Find the largest crowd at each time-stamp point. Each cluster can contain at least m number of objects so for finding distance between two consecutive clusters we will the distance between each pair of objects explained as

$$Dist[(o_i)_t, (o_j)_{t+1}], \text{ where } i, j \text{ belongs to 0 to obj-1.}$$

Select the lowest distance and then compare it with threshold distance δ. If distance between two consecutive clusters is found to be less than threshold distance δ then both cluster will be a part of crowd.

Algorithm for Crowd: in two stapes

2.3 Discovering Gathering from Crowd

Till Now we have already found crowd [8] to find the gathering. All the crowd which have at least m_p number of participator will be a part of gathering.

Step 1: Creating link step and checking threshold lifetime k_c

```
1: Input: C_DB, k_c, δ
2: link ← Null
3: For t_k = 0 to t-1 do
4:          c_i ← take each cluster from cluster set at time t_k  do
5: t' ← k+1
6:                 z←0
7:                      c_j ←take each cluster from cluster set at time t_k+1 do
8:        dist←Dist(c_i, c_j)
9:                          if(dist<= δ)
10:                         link [t_k][i][z++]←j
11:                       end
12:             end
13:          for i=0 to obj-1 do
14:             z←0
15: count←0
16:                 while link[t_k][i][z] is found do
17:                 count++
18:                 end
19:             if(count<k_c)
20:              delete all values from i^th row
21:          end
22: end
```

Step 2: detecting crowd from link

```
1: Input: C_DB, link
2: crowd ← Null
3: for t_k = 0 to (t-k_c) do
4:        For i=0 to obj-1 do
5:                z←0
6:                count←0
7:                k'←t_k
8:                  z'←0
9:                while link [t_k][i][z] is found do
10:                     k'++
11:                   crowd [t_i][z'++] ← cluster in z^th row at time k'
12:                 end
13:         end
14: end
```

2.3.1 Participator

Given in a crowd *Cr*, an object o is called a participator of *Cr* iff it appears in at least k_p snapshot clusters of *Cr*. Let Cr(o) denote the set of snapshot clusters in *Cr* that contains object o, i.e., *Cr* (o) = $\{c_t | c_t \in Cr, o(t) \in c_t\}$.

Then the participators of *Cr* are the object set *Par* (*Cr*) = $\{o \,||Cr(o)| \geq k_p\}$.

Example of Participator

Consider a closed crowd as shown in table 1, and let $k_c = k_p = 3$, $m_c = m_p = 3$. Let the 2 crowds are $Cr_a = c_1, c_2, c_3, c_4$ and $Cr_b = c_6, c_7, c_8$. According to the rule each cluster of both the crowd should have $m_p = 3$ no. of objects at least $k_p = 3$ times in other cluster of same crowd.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
	o_1	o_1		o_1	o_1		
o_2	o_2	o_2	o_2			o_2	o_2
o_3	o_3		o_3		o_3	o_3	o_3
o_4		o_4	o_4	o_4	o_4	o_4	o_4
	o_5	o_5	o_5				
				o_6	o_6		

Figure 1. Cluster with objects

In $Cr_a = c_1, c_2, c_3, c_4$ for c_1 having 3 participator o_2, o_3, o_4, same c_2 having 3 participator o_2, o_3, o_5 same c_3 having 3 participator o_2, o_4, o_5, and c_4 having 3 participator o_2, o_3, o_5 so $Cr_a = c_1, c_2, c_3, c_4$ is a gathering but $Cr_b = c_6, c_7, c_8$ is not a gathering because Cr_b is not following the condition for being gathering.

2.3.2 Algorithm for Gathering

1: input: crowd, k_p, m_p, k_c
2: R←∅ // set of gathering
3: for $t_k = 0$ to *t*- do
4: for *i* = 0 to obj−1 do
5: *z*←0
6: if C_{Ri} from crowd is found **then**
7: **if** check participator(C_{Ri}, k_p, m_p) is true **then**
8: $R[t_k][z{+}{+}] \leftarrow C_{Ri}$
9: end
10: end

2.4 Discovering Gathering Incrementally

We have discussed the efficient algorithms for discovering closed gatherings in a trajectory archive. But in real applications, trajectories are often received incrementally. As such, the latest batch of trajectory data should be appended to the database periodically (e.g., every day, week or month). Specifically, consider a trajectory database ODB with the time domain $T_{DB} = t_1; t_2; \ldots. t_n$. After a new batch of trajectories One with the time domain $T_{new} = t_{n+1} \ldots.. t_u$ has been collected and appended to O_{DB}, we obtain an updated database $O_{DB} = O_{DB} \cup O_{new}$ with the extended time domain $T_{DB} = T_{DB} \cup T_{new}$.

Now from here we get a problem that some crowds which ware closed crowd in old data that may be extended in new data. To get the correct result after new data incremented we are proposing some points.

1) All those crowds which has no clusters at time t_n they can't be extended because crowds are the set of clusters at consecutive timestamp.

2) Only those crowds we have to check to be extend which has cluster at t_n.

3) We have to check those non-crowds also from time between (t_n-k_c)(here k_c=lifetime) and t_n if it has cluster at t_n.

4) We have to check crowd for the extended time interval.

After checking these four terms in new database we can find the final extended crowd. Once the final super crowd will be found then we will go for the gathering and repeat the process for gathering as explained above.

3. Conclusion

In this paper we study the concept of discovering gathering patterns from a large-scale trajectory database. Here we see how gathering is different than crowd. And after finding cluster database we produced an algorithm for efficiently finding the crowd from cluster database during the given time interval. We see here if the data are incremented after given time then how we will handle it and update gathering.

References

[1] Zheng, Y., Zhou, X. (2011). *Computing with Spatial Trajectories*. New York, NY, USA: Springer.

[2] Ester, M., Kriegel, H., Sander, J., Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, *In*: Proceedings SIGKDD, 96, 226–231.

[3] Ester, M., Kriegel, H., Sander, J., Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, in SIGKDD, 96, 226–231.

[4] Jeung, H., Yiu, M., Zhou, X., Jensen, C., Shen, H. (2008). Discovery of convoys in trajectory databases, *VLDB Endow.*, 1, no. 1, 1068–1080, August.

[5] Lee, J., Han, J., Whang, K. (2007). Trajectory clustering: A partition-and-group framework, *In*: Proceedings SIGMOD, Beijing, China, 2007, a. 604.

[6] Rote, G. (1991). Computing the minimum Hausdorff distance between two point sets on a line under translation, *Inform. Process. Lett.*, 38 (3) 123–127.

[7] Rote, G. (1991). Computing the minimum hausdorff distance between two point sets on a line under translation, I*nformation Processing Letters*, 38, (3) 123–127.

[8] Zheng, K., Zheng, Y., Yuan, N. J., Shang, S. (2013). On discovery of gathering patterns from trajectories, *In*: *Proceedings ICDE*, Brisbane, QLD, Australia, 242–253.