

# Discovering Frequent Mobility Patterns on Moving Object Data

Ticiana L. Coelho da  
Silva  
Federal University of Ceará,  
Brazil  
ticianalc@ufc.br

José A. F. de Macêdo  
Federal University of Ceará,  
Brazil  
jose.macedo@lia.ufc.br

Marco A. Casanova  
Department of Informatics -  
PUC-Rio, Brazil  
casanova@inf.puc-rio.br

## ABSTRACT

We consider the problem of efficiently discovering and detecting frequent mobility patterns on moving object data. Our proposed approach is key for mobility applications, such as applications that need to discover and explain movement patterns of a set of moving objects (e.g. traffic management, birds migration, disease spreading). In this sense, we developed a method that performs density based clustering on trajectory data at regular time intervals, then we analyze clusters evolution, which is characterized by appear, disappear, expand, shrink, split, merge and survive. To solve our problem, a tree-based representation called Tree Evolution Cluster over Time ( $T_{ec}$ ) is described and an algorithm to generate the tree is also presented. Finally, we map our problem to the problem of discovering frequent tree paths on that tree. Therefore, the frequent tree paths are the frequent sequence of evolution patterns that occurs in the dataset. We discuss a preliminary solution to this problem and present some experimental results. The results suggest that evolution patterns and their frequency can be effectively obtained through the proposed  $T_{ec}$  obtained from moving object data.

## Categories and Subject Descriptors

H.2 [Database Management]: Miscellaneous; H.3 [Information Storage and Retrieval]: Miscellaneous; I.7 [Document and Text Processing]: Miscellaneous

## General Terms

Algorithm

## Keywords

Frequent pattern, Evolution, Clustering, Moving Object

## 1. INTRODUCTION

Mobility data has been fostered by the widespread diffusion of wireless technologies, such as the call detail records from mobile phones and GPS tracks from navigation devices, to

name two. These data open new opportunities to exploit knowledge of object movement for purposes such as targeted sales, traffic congestion prediction and animal migration, for example. Information about moving objects becomes increasingly available, which has posed new challenges to database research.

Imagine the following scenario, illustrated in Figure 1(a): at time  $t$ , there are three clusters and some outliers on the dataset. Consider that each point on the dataset is a moving object and each cluster represents a density-area, for example. After  $\delta t$  units of time, the objects of clusters  $C_1$  and  $C_2$  moved to the same space and they are merged to only one cluster. Observe that on Figure 1, we label to cluster  $C_1$  the objects that belonged to cluster  $C_2$  on the previous time. This phenomenon is called cluster evolution pattern and it can happen on a real scenario (imagine a highway that have two roads which merge at some point). Note that, for cluster  $C_3$ , more moving objects were added to the dataset between  $t + \delta t$  and  $t + 2\delta t$ . Cluster  $C_3$  at time  $t + 2\delta t$  can represent, for example, the increase of vehicles in an area and it may represent a congested area.

The cluster evolution of moving objects should take into accounting objects previous movements, their updated movement as time goes by, including the deletion and insertion of others moving objects. As we shall see, doing so enables us to capture each clustering change as it occurs during the continuous motion process. Thus, with aims tracking the cluster evolution patterns at each moment from such dynamic dataset. Typical cluster evolution patterns include appear, disappear, expand, shrink, split, merge and survive [?].

Many successful and scalable approaches have been proposed [?, ?, ?, ?, ?, ?, 3], which have achieved success on clustering static mobility datasets. They answer the question "Where are the moving object clusters?". However, in many scenarios, users may want to know more details about an event and may like to submit advanced queries like "How moving object clusters evolve over the geographical space?". This question is solved using the approaches proposed on [?, 9, 12]. In this paper, we analyse cluster evolution and address queries like "Which are the frequent patterns of cluster evolution on moving objects trajectories?". For example, to analyse traffic behavior, existing event detection approaches can discover where the congested areas are within the city at each moment, but they can not answer queries like "how

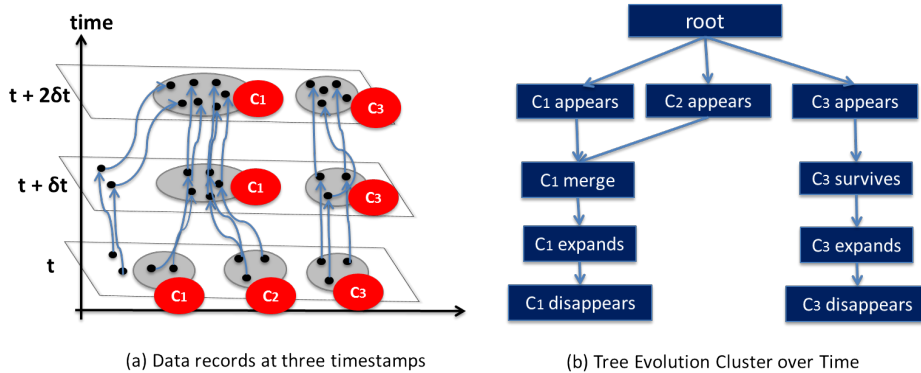


Figure 1: Example of detecting and discovering frequent cluster evolution pattern

do these congested areas interact with each other over time throughout the observed time?”.

In this paper we focus on modeling cluster evolution patterns, the evolution and interaction of these clusters over time, and the frequent sequence evolution patterns on the moving object data. Discovering frequent patterns does not reduce to counting how many times a pattern occurs on the dataset. It is computationally costly to calculate the frequency for every possible sequence of cluster evolution on the dataset.

On Section 3 we present our approach, which does not require to check all the combinations of possible evolutions to discover which are the most frequent. Furthermore, we present a naive approach to detect cluster evolution patterns, considering that the clustering algorithm is executed over all dataset. It is worth noting that this approach is not optimal since we will compute clusters at every  $\delta t$  units of time and for each execution we will evaluate the cluster evolutions. After, we discover the frequent sequences of cluster evolution.

Our proposed approach is key for mobility applications, for instance, applications that need to discover mobility pattern on trajectory data. A mobility pattern represents the common behaviour of a group or subgroup of trajectories, obtained as a result of a data mining algorithm. For example, to find gathering on trajectory data [?]. Informally, a gathering represents a group event or incident that involves congregation of objects (e.g., vehicles, people, animals). Examples of gatherings may include celebrations, parades, large-scale business promotions, protests, traffic jam.

For example, we may describe a traffic jam as a sequence of evolution patterns as we show on Figure 1(a): two clusters  $C_1$  and  $C_2$  appear at time  $t$ . After  $\delta t$  units of time,  $C_1$  and  $C_2$  move closer to each other and they merge into one cluster, we label it as  $C_1$ . At  $t + 2\delta t$ , imagine that  $C_1$  expands if the traffic jam gets worse. Or  $C_1$  shrinks, otherwise. When the traffic jam is over, the cluster evolution of  $C_1$  is disappear. If this sequence (*appear, merge, expand|shrink, disappear*) occurs frequently on the dataset, we claim that traffic jam is a frequent pattern. Therefore to solve our problem on this work, we model all cluster evolution on a Tree Evolution

Cluster over Time ( $T_{ec}$ ) as show in Figure 1(b).Then, we map our problem on discovering frequent tree paths on  $T_{ec}$ . The frequent tree paths are frequent sequences of evolution patterns that occurs on the dataset. We will discuss our solution on Section 3.

Many applications needs to cope with cluster changes in the mobility domain, among them traffic management, delivery logistics and crowd monitoring, it is also necessary to provide insights about the nature of cluster change: Is a cluster corresponding to a group of vehicles simply disappearing or are its members migrating to other clusters? Is a new emerging cluster reflecting a new target group of vehicles or does it rather consist of existing customers whose preferences shift? [?].

Unlike others existing approaches, our approach does not handle only static datasets [?, ?, ?, ?, ?, ?, 3, ?, ?]. Our approach observes all cluster evolution patterns, unlike the approach described [?, 9, 13].The papers [?] and [12] are technically related to our work, considering cluster evolution patterns and dynamic data, however the first one works only for social networks and both of them do not take into account to find frequent evolution patterns. Moreover, the updated movement of moving objects as time goes by as we do in our approach.

The paper is organized as follows: Section 2 presents some definitions and the problem statement. The Section 3 discusses our solution and Section 4 describes our experimental study and the importance of our solution. Section 5 presents our related work. Finally, Section 6 contains the conclusion and future work.

## 2. DEFINITIONS

In this section, we start by defining the basic notions of moving object, moving object position and consistent set of moving object positions.

**DEFINITION 1. (Moving object)** A moving object is a pair  $\omega = (oid, \rho)$  such that *oid* is a unique identifier for  $\omega$  and  $\rho(t) : \mathbb{R} \rightarrow \mathbb{R}^2$  is the trajectory of  $\omega$  such that  $\rho(t) = (lat, lon)$  maps each point in time  $t \in \mathbb{R}$  into a pair  $(lat, lon) \in \mathbb{R}^2$ .

**DEFINITION 2. (Moving object position)** Let  $\omega = (oid, \rho)$  be a moving object and  $t \in \mathfrak{R}$ . The position of  $\omega$  at time  $t$  is the triple  $= (oid, \rho(t), t)$ .

**DEFINITION 3. (Consistent set of moving object positions)** Let  $\Omega$  be a set of moving objects. Let  $O$  be a set of positions of moving objects in  $\Omega$ . Then,  $O$  is consistent iff, for any  $(i, p, t), (j, q, u) \in O$ ,  $i \neq j$  and  $t = u$ .

We now focus on the evolution of a set of moving object positions that results from changes in the objects positions and from the deletion and insertion of moving objects. These changes are modeled as events that represent the execution of one of the operations in the next definitions.

In what follows, let  $\Omega$  be a set of moving objects and  $O^\Omega$  be the set of all consistent sets of moving object positions of moving objects in  $\Omega$ .

**DEFINITION 4. (Insertion operation)** The insertion operation  $o^+ : \mathfrak{R} \times \Omega \times O^\Omega \rightarrow O^\Omega$  is such that, for each time  $t \in \mathfrak{R}$ , each moving object  $\omega = (oid, rho) \in \Omega$  and each set  $O$  of positions of objects in  $\Omega$  at time  $t$ ,  $o^+(t, \omega, O) = P$  iff  $P = O \cup \{(oid, \rho(t), t)\}$  (if  $O$  already contains the position of  $\omega$  at  $t$ , then  $O$  remains unchanged).

**DEFINITION 5. (Deletion operation)** The deletion operation  $o^- : \mathfrak{R} \times \Omega \times O^\Omega \rightarrow O^\Omega$  is such that, for each time  $t \in \mathfrak{R}$ , each moving object  $\omega = (oid, rho) \in \Omega$  and each set  $O$  of positions of objects in  $\Omega$  at time  $t$ ,  $o^-(t, \omega, O) = P$  iff  $P = O - \{(oid, \rho(t), t)\}$  (if  $O$  does not contain the position of  $\omega$  at  $t$ , then  $O$  remains unchanged).

**DEFINITION 6. (Update operation)** The update operation  $o^* : \mathfrak{R} \times O^\Omega \rightarrow O^\Omega$  is such that, for each time  $t \in \mathfrak{R}$  and each set  $O$  of positions of objects in  $\Omega$  at time  $t$ ,  $o^*(t, O) = P$  iff  $P = \{(oid, \rho(t), t) / (\exists(i, \rho) \in \Omega)(\exists u \in \mathfrak{R}) ((i, \rho(u), u) \in O \wedge oid = i)$ .

**DEFINITION 7. (Event)** An event is a call to an insertion, deletion or update operation with a specific set of inputs.

We now turn to concepts related to clustering of moving object positions.

Clustering is a major data mining technique that groups a set of objects in such a way that objects in the same group are more similar to each other than to those in different groups. To discover cluster evolution patterns, we first need to apply a clustering algorithm to our dataset. A spatial cluster is a group of entities in spatial proximity. Various clustering algorithms can be used, including partitioning based, hierarchical, model based, and density based. We argue that a density-based clustering is better suitable to our work due to the following properties: (1) the ability to construct non-spherical clusters of arbitrary shapes; (2) the robustness

with respect to noise in the data; (3) the ability to discover an arbitrary number of clusters without pre-specifying the number of clusters.

We first define the Jaccard similarity between two sets of consistent moving object positions, which will be used in Algorithm 1 on Section 3.2 to detect the evolution patterns of sets of moving objects.

Given  $O \in O^\Omega$ , let  $O_{obj}$  denote the set of moving objects with positions in  $O$ :

$$O_{obj} = \{oid / (\exists(oid, \rho) \in \Omega)(\exists t \in \mathfrak{R})(oid, \rho(t), t) \in O\}$$

**DEFINITION 8. (Jaccard Similarity)** Let  $O, P \in O^\Omega$ . The Jaccard Similarity of  $O$  and  $P$  is defined as:

$$Similar(O, P) = \frac{\|O_{obj} \cap P_{obj}\|}{\|O_{obj} \cup P_{obj}\|}$$

The following definition introduces the notion of a cluster of moving object positions, omitting the details of the clustering method.

**DEFINITION 9. (Clustering of Moving Object Positions)** Let  $O$  be a consistent set of moving object positions. A clustering of  $O$  is a partition  $C_t = \{S_0, \dots, S_m, Out\}$  of  $O$  where  $Out$  is called the set of outliers.

In what follows, let  $\Omega$  be a set of moving objects and  $C^\Omega$  be the set of all clusterings of consistent sets of moving object positions of moving objects in  $\Omega$ . As a convenience, we assume that the empty set  $\emptyset$  is in  $C^\Omega$ .

The next definition introduces the concept of a cluster evolution, which is used to model our approach to discover frequent evolution patterns. Let  $EP = \{‘expands’, ‘shrinks’, ‘survives’, ‘disappears’, ‘merges’\}$  be a set of actions (see Section 3.1).

**DEFINITION 10. (Clustering Evolution)** A clustering evolution over  $\Omega$  is a labeled directed graph  $\Delta_{evol} = (N, E, \Lambda)$  such that

- $\Delta_{evol}$  is acyclic and has a single source, called the root of  $\Delta_{evol}$
- $\Lambda$  assigns a label of the form  $(p, O, t)$  to each node in  $N$ , except for the root, where  $p \in EP$  and  $O$  is a set of positions of moving objects in  $\Omega$  at time  $t \in \mathfrak{R}$ ; and  $\Lambda$  labels the root with empty set
- A node in  $N$  has an indegree greater than 1 iff it is labelled with ‘merge’; otherwise the node has indegree equal to 1
- A node in  $N$  has an outdegree equal to 0 iff it is labelled with ‘disappear’; otherwise the node has outdegree equal to 1

**Table 1: Notation**

Notation	Meaning
$S_i$	a cluster
$C_t$	a set of clusters $\{S_0, \dots, S_m, Out\}$ at time $t$
$O_t$	dataset of moving object at time $t$
$Similar(S_i, S_j)$	Similarity function to detect cluster evolution
$T_{ec}$	Tree Evolution Cluster over Time
$H_i$	a set with all the frequent subgraphs on $T_{ec}$ with cardinality equals to $i$
$F$	set of frequent subgraphs $\{H_1, \dots, H_k\}$ on $T_{ec}$
$\sigma$	minimum support to detect frequent patterns

Figure 1 shows an example of a clustering evolution  $\Delta_{evol}$ . Note that  $\Delta_{evol}$  is almost a tree, except for the nodes whose labels contain ‘merge’. Indeed, the label  $(p, O, t)$  of a node indicates that cluster  $O$  was the result of applying action  $p$  to one or more clusters. Note that a cluster in a clustering expands, shrinks, survives, merges or disappears into just one cluster; therefore, the outdegree of all nodes is equal to 1. Furthermore, a cluster in a clustering may be the result of a merge of other clusters; in this case, and only in this case, the indegree of the node is greater than 1.

To calculate how frequent an evolution pattern is, we model each pattern as a subgraph  $H$  of  $\Delta_{evol}$ . Then, we find all subgraphs of  $T_{ec}$  which are isomorphic to  $H$ . If the number of all such subgraphs is greater than a threshold  $\sigma$ , we consider that  $H$  is a frequent pattern.

**Problem Statement** Let  $O_t$  be a consistent set of moving object positions of moving objects that belongs to  $\Omega$  at time  $t$ . Let  $C_t$  be a clustering of  $O_t$ . As we consider that our dataset is dynamic, after  $\delta t$  units of time,  $O_t$  can change to  $O_{t+\delta t}$  due to the operations  $U_{t+\delta t}$  previously defined. If  $U_{t+\delta t} = \emptyset$ , the set  $C_t$  does not change. The input is  $O_t, U_{t+\delta t}, C_t$ , and the procedure goes as follows:

1. Discover and detect the cluster evolution of each  $S_i \in C_t$  at time  $t + \delta t$
2. Generate  $T_{ec}$  over time as the Definition 10
3. Find the frequent cluster evolution patterns  $F = \{H_1, \dots, H_k\}$  on  $T_{ec}$  with respect to support  $\sigma$

The output is  $C_{t+\delta t}$ ,  $\forall S_i \in C_t$  the output also specifies the  $S_i$  evolution pattern on  $C_{t+\delta t}$  and a set with the frequent evolution patterns  $F = \{H_1, \dots, H_k\}$ , such that  $\forall i \in \{1, \dots, k\}, H_i$  is a set with all the paths with cardinality equals to  $i$  on  $T_{ec}$  that occurs more than a support  $\sigma$ .

### 3. PROPOSED SOLUTION

Since we propose a cluster evolution approach for moving object datasets, we need to monitor all cluster evolution patterns. With this idea in mind, we map our solution to ECA(Event Condition Action) rules. As already defined, an

event is the execution of an operation. A condition represents all the predicates that must be met so that an event happens. If the condition is true, it results in an action that is a cluster transition from  $t$  to  $t + \delta t$ . These conditions are used on Algorithm 1 to find the clusters evolution patterns.

Table 1 describes notation used throughout the paper.

### 3.1 Definitions of Actions and Conditions

In this section, we define the actions, which represents cluster transitions (evolution) from  $t$  to  $t + \delta t$ . In each action/transition below, the enumerated sentences must be true. Assume that at time  $t$  the cluster of moving objects is  $C_t = \{S_0, \dots, S_m, Out\}$ . At time  $t + \delta t$ , after operation  $U_{t+\delta t}$ ,  $C_{t+\delta t} = \{S'_0, \dots, S'_m, Out'\}$ . Let  $\tau \in (0, 1]$  (in our experiments, we discuss how to define this threshold).

1. **Action:**  $S_i \Rightarrow S'_j$   
(Cluster  $S_i \in C_t$  survives as cluster  $S'_j \in C_{t+\delta t}$ ).  
**Condition:**  $Similar(S'_j, S_i) = 1$
2. **Action:**  $\{S_{i_1}, \dots, S_{i_k}\} \Rightarrow S'_j$   
(Two or more clusters  $S_{i_1}, \dots, S_{i_k} \in C_t$  are merged into a single cluster  $S'_j \in C_{t+\delta t}$ ).  
**Conditions:**
  - (a)  $Similar(S_{i_1} \cup \dots \cup S_{i_k}, S'_j) \geq \tau$
  - (b)  $\forall S_r \in C_t \setminus \{S_{i_1}, \dots, S_{i_k}\}, Similar(S_{i_1} \cup \dots \cup S_{i_k} \cup S_r, S'_j) < \tau$
3. **Action:**  $S_j \Rightarrow \{S'_{i_1}, \dots, S'_{i_k}\}$   
(Cluster  $S_j \in C_t$  splits into two or more clusters  $S'_{i_1}, \dots, S'_{i_k} \in C_{t+\delta t}$ ).  
**Conditions:**
  - (a)  $Similar(S'_{i_1} \cup \dots \cup S'_{i_k}, S_j) \geq \tau$
  - (b)  $\forall S'_r \in C_{t+\delta t} \setminus \{S'_{i_1}, \dots, S'_{i_k}\}, Similar(S'_{i_1} \cup \dots \cup S'_{i_k} \cup S'_r, S_j) < \tau$
4. **Action:**  $S_i \nearrow$   
(Cluster  $S_i \in C_t$  expands w.r.t. its size).  
**Condition:**  $\exists S'_j \in C_{t+\delta t}$  such that  $S_i \subset S'_j$  (that is,  $S_i$  is a strict subset of  $S'_j$ ).
5. **Action:**  $S_i \searrow$   
(Cluster  $S_i \in C_t$  shrinks w.r.t. its size).  
**Condition:**  $\exists S'_j \in C_{t+\delta t}$  such that  $S'_j \subset S_i$  (that is,  $S'_j$  is a strict subset of  $S_i$ ).
6. **Action:**  $S_i \Rightarrow \odot$   
(Cluster  $S_i \in C_t$  disappears).  
**Condition:**  $\forall S'_j \in C_{t+\delta t}, Similar(S_i, S'_j) < \tau$
7. **Action:**  $\odot \Rightarrow S'_i$   
(Cluster  $S'_i \in C_{t+\delta t}$  appears).  
**Condition:**  $\forall S_j \in C_t, Similar(S_j, S'_i) < \tau$

Table 2 shows the events that may result in each action explained above. We relate them using ‘X’ in the cells of Table 2. When an event may trigger an action, there are conditions that must be checked. These conditions are defined

**Table 2: Events that may result on Actions**

Action \ Event	$o+$	$o-$	$o*$
$S_i \Rightarrow S'_j$	-	-	-
$\{S_{i_1}, \dots, S_{i_k}\} \Rightarrow S'_j$	<b>X</b>	-	<b>X</b>
$S_j \Rightarrow \{S'_{i_1}, \dots, S'_{i_k}\}$	-	<b>X</b>	<b>X</b>
$S_i \nearrow$	<b>X</b>	-	<b>X</b>
$S_i \searrow$	-	<b>X</b>	<b>X</b>
$S_i \Rightarrow \odot$	-	<b>X</b>	<b>X</b>
$\odot \Rightarrow S'_i$	<b>X</b>	-	<b>X</b>

on each action above. In the next subsection, we will present a naive approach to detect cluster evolution patterns using the conditions above, considering that the clustering algorithm is executed over all dataset. It is worth noting that this approach is not optimal since we will compute clusters at every  $\delta t$  units of time and, for each execution, we will evaluate the cluster evolutions.

### 3.2 Naive Approach

Considering  $C_t$  as all the clusters found at time  $t$  and  $C_{t+\delta t}$  as all clusters found at time  $t + \delta t$ , we propose a naive strategy (Algorithm 1) to detect the evolution from  $C_t$  to  $C_{t+\delta t}$ . This algorithm also generates  $T_{ec}$  incrementally.

In our approach we resort to a matrix  $M$  that stores in each cell the similarity value (computed according Definition 8) between two clusters in different timestamps. In this matrix, the lines represent the clusters that belong to  $C_t$  and the columns represent the clusters that belong to  $C_{t+\delta t}$ . For example,  $M[S_i, S'_j]$  corresponds to the Jaccard similarity of the cluster  $S_i \in C_t$  to cluster  $S'_j \in C_{t+\delta t}$ .

While the matrix  $M$  is calculated, the Algorithm 1 also generates  $T_{ec}$ . For each cluster  $S_i \in C_t$ , the algorithm gets the vertex  $u$  on  $T_{ec}$  that corresponds to the cluster  $S_i$ . If  $\exists S'_j \in C_{t+\delta t}$ , such that  $M[S_i, S'_j] \geq \tau$ , then the evolution of  $S_i$  to  $S'_j$  can be survive, split, merge, disappear, shrink or expand. A vertex  $v$  is created to represent  $S'_j$  and an edge  $\{u, v\}$  is also added on  $E(T_{ec})$ . On the case of appear evolution pattern, a cluster  $S'_j \in C_{t+\delta t}$  appears if  $\forall S_i \in C_t, M[S_i, S'_j] = 0$ . Therefore, a vertex  $v$  is created for  $S'_j$  and an edge  $\{root, v\}$  is added from  $root$  to  $v$  on  $E(T_{ec})$ .

Algorithm 1 receives as input  $M$  and detects each cluster evolution pattern from time  $t$  to time  $t + \delta t$ . The complexity of this naive algorithm is  $O(n^2)$ , assume that  $n$  is the order of the number of clusters. This processing can be costly, moreover it needs to execute the clustering algorithm from scratch. On our future approach we aim to detect cluster evolution using operations insert, delete or update moving object at each  $\delta t$  units of time, without recomputing all clusters.

A drawback of this Naive approach is to define an adequate value for  $\tau$  that represents the similarity among the clusters. Anyway this algorithm is useful for our study, since it represents a brute force approach and may reveal the challenges we will face in our work. The next section presents our experimental evaluation using this naive approach.

---

#### Algorithm 1: Find each cluster evolution pattern

---

**Input:** Similarity matrix  $M$ ,  $C_t, C_{t+\delta t}, \tau, T_{ec}$

```

1 begin
2   if  $T_{ec} = \emptyset$  then
3     root  $\leftarrow$  newVertex(root, 0, t)
4      $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{root\}$ 
5   for  $S_i \in C_t$  do
6      $u \leftarrow$  getVertex( $T_{ec}, sid_i$ )
7     if  $\exists S'_j \in C_{t+\delta t}$  and  $M[S_i, S'_j] = 1$  then
8        $S_i$  survives
9        $v \leftarrow$  newVertex(survives,  $sid_i, t + \delta t$ )
10       $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
11       $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
12    else
13      if  $\exists \{S'_k \cup \dots \cup S'_j\} \subseteq C_{t+\delta t}, \sum_{r=k}^j M[S_i, S'_r] \geq \tau$ 
14        then
15           $S_i$  splits
16          for  $S'_r \in \{S'_k \cup \dots \cup S'_j\}$  do
17             $v \leftarrow$  newVertex(split,  $sid_r, t + \delta t$ )
18             $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
19             $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
20        else
21          if  $\exists S'_j \in C_{t+\delta t}, \exists \{S_i \cup \dots \cup S_k\} \subseteq$ 
22             $C_t, \sum_{r=i}^k M[S_r, S'_j] \geq \tau$  then
23             $S_i, \dots, S_k$  merge
24             $v \leftarrow$  newVertex(merge,  $sid_j, t + \delta t$ )
25             $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
26             $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
27          else
28            if  $\nexists S'_j \in C_{t+\delta t}, M[S_i, S'_j] \geq \tau$  then
29               $S_i$  disappears
30               $v \leftarrow$  newVertex(disappear,  $sid_{-1}, t + \delta t$ )
31               $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
32               $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
33            else
34              if  $\exists S'_j \in C_{t+\delta t}, M[S_i, S'_j] \geq \tau$  then
35                if  $|O_{S_i}| > |O_{S'_j}|$  then
36                   $S_i$  shrinks
37                   $v \leftarrow$  newVertex(shrink,  $sid_j, t + \delta t$ )
38                   $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
39                   $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
40                if  $|O_{S_i}| < |O_{S'_j}|$  then
41                   $S_i$  expands
42                   $v \leftarrow$  newVertex(expand,  $sid_j, t + \delta t$ )
43                   $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
44                   $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{u, v\}$ 
45          for  $S'_j \in C_{t+\delta t}$  do
46            if  $\forall S_i \in C_t, M[S_i, S'_j] = 0$  then
47               $S'_j$  appears
48               $v \leftarrow$  newVertex(appear,  $sid_i, t + \delta t$ )
49               $V(T_{ec}) \leftarrow V(T_{ec}) \cup \{v\}$ 
50               $E(T_{ec}) \leftarrow E(T_{ec}) \cup \{root, v\}$ 

```

---

**Algorithm 2:** Apriori-based approach to detect frequent sequence evolution

**Input:**  $T_{ec}$ : a Tree Evolution Cluster,  $\sigma$ : minimum support  
**Output:**  $H_1, \dots, H_k$  a set of frequent subgraphs of cardinality 1 to  $k$

```

1 begin
2    $H_1 \leftarrow$  detect all frequent paths with one vertex in  $T_{ec}$ 
3    $k \leftarrow 2$ 
4   while  $H_{k-1} \neq \emptyset$  do
5      $H_k \leftarrow \emptyset$ 
6      $CH_k \leftarrow$  candidate - gen( $H_{k-1}$ )
7     for  $g \in CH_k$  do
8        $SI \leftarrow$  path-isomorphism( $g, T_{ec}$ )
9        $g.count \leftarrow |SI|$ 
10      if  $g.count \geq \sigma \wedge g \notin H_k$  then
11         $H_k \leftarrow H_k \cup g$ 
12     $k \leftarrow k + 1$ 

```

After generate  $T_{ec}$ , we need to find all frequent sequence evolution patterns with cardinality varying on 1 to  $k$ . This is the same of discovering the frequent paths on  $T_{ec}$ . For that, we used apriori-based approach discussed on [?]. This algorithm is presented on Algorithm 2. Observe that we can identify the frequent pattern sequences involving three or more time instances using  $k \geq 3$ .

The basic Apriori-based algorithm is presented [?]. In line 6 all frequent ( $k-1$ ) path are used to generate  $k$  path candidates. If any of the  $k-1$  candidate path are not frequent, then the algorithm safely prunes the candidates. For each candidate  $g$ , the algorithm keeps on  $SI$  all the isomorphic path to  $g$  on  $T_{ec}$  (line 8). If the number of isomorphic path is greater than the support  $\sigma$ ,  $g$  is a candidate of frequent path or sequence evolution pattern. So that,  $g$  is stored on  $F_k$ .

There are many approaches to efficiently find the isomorphic path to  $g$  on  $T_{ec}$ . They are also presented on [?].

#### 4. EXPERIMENTAL EVALUATION

The dataset used in the experiments were related to vehicles trajectories from Fortaleza city in Brazil, and it contains for each vehicle a collection of timestamped positions (latitude and longitude) obtained from a GPS device. Each vehicle represent one moving object on our experiments. In these experiments we run the most famous density based clustering algorithm DBScan [?] from scratch at every  $\delta t$  units of time. We used  $\delta t$  equals to one hour, the DBScan parameters are  $eps = 50$  and  $minPoints = 5$ . We defined  $\tau$  equals to 0,5.

In these experiments, our main goal is to show for all clusters how to detect on every  $\delta t$  units of time its evolution pattern. Moreover, to discover all the frequent sequence of cluster evolution pattern. We believe that a sequence of evolution can describe a mobility pattern as we did before for modeling traffic jam in the Introduction part.

The number of trajectories used on each execution of DBScan is described on Table 3 as well as the number of clus-

**Table 3: Number of trajectories and clusters**

Timestamp	#Trajectories	#clusters
20:00	1314	18
21:00	1308	27
22:00	1306	38

**Table 4: Some clusters evolution patterns**

Matrix Element(s)	Evolution Pattern
<b>Evolution from 20:00 to 21:00</b>	
$M[S_1, S'_1]=1$	$S_1$ survives
$M[S_2, S'_3]=0,75, \forall S_i \in C_{20:00} i \neq 2,$ $M[S_i, S'_3] = 0$ and $ O_{S_2}  >  O_{S'_3} $	$S_2$ decreases
$M[S_5, S'_{17}]=1$	$S_5$ survives
$M[S_9, S'_{13}]=0,46$ and $M[S_{11}, S'_{13}]=0,1$	$S_9$ and $S_{11}$ merge
$\forall S_i \in C_{21:00}, M[S_i, S'_{25}] = 0$	$S'_{25}$ appears
<b>Evolution from 21:00 to 22:00</b>	
$M[S_1, S'_3]=1$	$S_1$ survives
$M[S_2, S'_4]=0,83, \forall S_i \in C_{21:00} i \neq 2,$ $M[S_i, S'_4] = 0$ and $ O_{S_2}  <  O_{S'_4} $	$S_2$ increases
$\forall S'_j \in C_{22:00}, M[S_5, S'_j]=0$	$S_5$ disappears
$M[S_{12}, S'_7]=0,06$ and $M[S_{12}, S'_{15}]=0,62$	$S_{12}$ splits
$M[S_{13}, S'_{16}]=0,57$ and $M[S_{13}, S'_{19}]=0,14$	$S_{13}$ splits
$M[S_{15}, S'_{16}]=0,05$ and $M[S_{15}, S'_{19}]=0,57$	$S_{15}$ splits
$M[S_{24}, S'_{25}]=1$	$S_{24}$ survives
$\forall S_i \in C_{21:00}, M[S_i, S'_1] = 0$	$S'_1$ appears

ters found. Note that the number of trajectories is almost the same in all cases, however there are new, deleted and updated trajectories and thus moving objects between two different timestamps.

In the Table 4 we present some cases of clusters evolution patterns found and the result covers all the patterns discussed before. Note that the clusters do not remain the same for different timestamps. For example, the cluster  $S_5 \in C_{21:00}$  disappears and the cluster  $S'_1$  appears on  $C_{22:00}$ . We did not show all the elements of the two obtained evolution matrix, because it will not be trivial to visualize.

The results on Table 5 present the frequent sequence of cluster evolution pattern found by our approach using  $k=5$  and  $\sigma=20$ . The results suggest that evolution patterns and their frequency can be effectively obtained through the proposed Tree Evolution Cluster from moving object data. Observe that merge and split evolutions are more frequently than others. Both cluster evolutions capture the evolvments and interaction of clusters over time as we aim.

In this context, our work is important because we aim to find the clusters evolution patterns and we proposed an approach to detect frequent mobility patterns using cluster evolution. Using this naive approach we have to execute the DBScan clustering algorithm from scratch on every  $\delta t$  units of time, which is time consuming. As future work, we aim to detect automatically cluster evolution patterns.

## 5. RELATED WORK

The works [?, ?, ?, ?, ?, ?, 3, ?, ?] proposed algorithms for three different clustering paradigms, i.e., k-partitioning, density-based and hierarchical clustering. However, all these solutions are clustering algorithm, these works do not discover evolution patterns neither frequent evolution patterns.

[?] proposes a novel adaptive framework for evolutionary clustering by performing tracking followed by static clustering. The objective of the framework is to accurately track the true proximity matrix at each time step, for that it uses a recursive update with an adaptive forgetting factor that controls the amount of weight to apply to historic data. The main advantage of that approach is its universality, allowing almost any static clustering algorithm to be extended to an evolutionary one. Moreover [?] is for static dataset and it does not find frequent evolution patterns.

**Table 5: Some Frequent Sequence Cluster Evolution Patterns**

Frequent Sequence Evolution Pattern	Frequency
<b>k=1</b>	
SHRINKS	331
EXPANDS	317
SPLIT	206
MERGE	178
<b>k=2</b>	
EXPANDS-SHRINKS	117
MERGE-MERGE	160
SPLIT-SPLIT	150
EXPANDS-EXPANDS	90
<b>k=3</b>	
MERGE-MERGE-MERGE	100
SPLIT-SPLIT-MERGE	62
SHRINKS-EXPANDS-SHRINKS	51
EXPANDS-SHRINKS-EXPANDS	45
EXPANDS-EXPANDS-SHRINKS	35
<b>k=4</b>	
MERGE-MERGE-MERGE-MERGE	46
SPLIT-SPLIT-SPLIT-MERGE	45
MERGE-MERGE-MERGE-EXPANDS	36
MERGE-MERGE-MERGE-SPLIT	34
SPLIT-SPLIT-MERGE-MERGE	33
<b>k=5</b>	
SPLIT-SPLIT-MERGE-MERGE-MERGE	41
SPLIT-SPLIT-SPLIT-MERGE-SPLIT	27
MERGE-MERGE-MERGE-MERGE-EXPANDS	26
SPLIT-SPLIT-SPLIT-SHRINKS-SPLIT	24
SPLIT-SPLIT-MERGE-MERGE-SPLIT	23

[9] proposes a fast and effective scheme for the continuous clustering of moving objects. It presents a dynamic summary data structure for clusters that enables frequent updates to the data without the need for global reclustering. An average-radius function is used that automatically detects cluster split events, which, in comparison to existing approaches [13], eliminates the need to maintain bounding boxes of clusters with large amounts of associated violation events. [9] and [13] considers dynamic data as our work, but they do not take into account all cluster evolution patterns mentioned before. Another difference is they do not find frequent evolution patterns on the dataset.

Another related work is [?]. It focuses on the cluster evolution tracking problem on highly dynamic networks. [?] takes the event evolution tracking task in social streams as an application, where a social stream and an event are modeled as a dynamic post network and a dynamic cluster respectively. [?] summarizes the network by a skeletal graph and monitor the updates to the post network by means of a sliding time window. It formalizes cluster evolution patterns using a group of primitive evolution operations and their algebra. Two incremental computation algorithms (split and merge clusters) are developed to maintain clusters and track evolution patterns as time rolls on and the network evolves. Unlike previous approaches, their evolution tracking algorithm performs incremental bulk updates in real time. [?] is similar to our work, considering that all cluster evolution patterns are observed. However, it works for social network, moreover it does not consider the update operation as we consider that a moving object can change its position as time goes by. That work does not present an approach to find frequent evolution patterns.

[12] proposes a framework that efficiently support the online discovery of moving objects that travel together. The framework adopts a sampling-independent approach, monitors clusters continuously and records the histories of clusters. The evaluation of groups is only dependent on the most recent history, so that the framework is amenable to online processing. [12] is very similar to our work, considering all cluster evolution patterns and defining the conditions that may lead to these transitions as we did in Section 3.1. However, [12] does not find frequent evolution patterns. Moreover, the updated movement of moving objects as time goes

by is not considered on the clustering algorithm as we do in our approach.

Considering trajectory data, [?] proposes new conceptual definitions for a spatio-temporal pattern named Herd and four types of herd evolvments: expand, join, shrink, and leave. These evolvments are similar to our expand, merge, shrink and disappear cluster evolution patterns. Herd evolvments are identified through measurements of Precision, Recall, and F-score. [?] finds spatio-temporal patterns using cluster evolution, this is similiar to our work because it is possible to define mobility patterns discovering frequent sequence evolution patterns. However [?] does not consider all evolution patterns and it does not present an approach to detect all of them.

Finally, [?] discovers patterns, called gathering, which is a trajectory pattern modelling various group incidents such as celebrations, parades, protests, traffic jams and so on. A key observation is that these incidents typically involve large congregations of individuals, which form durable and stable areas with high density. Since the process of discovering gathering patterns over large-scale trajectory databases can be quite lengthy, [?] further develop a set of well thought out techniques to improve the performance. These techniques, including effective indexing structures, fast pattern detection algorithms implemented with bit vectors, and incremental algorithms for handling new trajectory arrivals. However, it does not using cluster evolution patterns on its approach to find gathering (a mobility pattern).

## 6. CONCLUSION AND FUTURE WORK

We consider the problem of efficiently discovering cluster evolution and detecting frequent sequence of evolution patterns on moving object data. To solve this problem we consider the possible cluster transitions: appear, disappear, expand, shrink, split, merge and survive. We also defined a tree-based representation for clusters evolution patterns over time and we map our problem in discovering frequent subgraphs on the tree. Therefore, as a solution we have the most frequent sequences of clusters evolution patterns that occurs on the dataset. We discuss a preliminary solution to that problem and present some experimental results. The results suggest that evolution patterns and their frequency can be effectively obtained through the proposed Tree Evolution Cluster from moving object data.

As future work, we will present how the conditions defined in the Section 3.1 can be efficiently monitored. Thus, we do not need to execute clustering algorithm from scratch after  $\delta t$  units of time.

## 7. REFERENCES

- [1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. *ACM SIGMOD Record*, 28(2):49–60, 1999.
- [2] J. L. Bentley. Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, volume 18, pages 509–517. ACM, 1975.
- [3] T. L. Coelho da Silva, A. C. N. Araujo, R. P. Magalhaes, V. A. E. d. Farias, J. A. de Macedo, and J. C. Machado. Efficient and distributed dbscan algorithm using mapreduce to detect density areas on traffic data. ICEIS, 2014.
- [4] B.-R. Dai and I.-C. Lin. Efficient map/reduce-based dbscan algorithm with optimized data partition. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 59–66. IEEE, 2012.
- [5] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.
- [7] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *VLDB J.*, 20(5):695–719, 2011.
- [8] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 473–480. IEEE, 2011.
- [9] C. S. Jensen, D. Lin, and B.-C. Ooi. Continuous clustering of moving objects. *Knowledge and Data Engineering, IEEE Transactions on*, 19(9):1161–1174, 2007.
- [10] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.
- [11] S. Kisilevich, F. Mansmann, and D. Keim. P-dbscan: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, page 38. ACM, 2010.
- [12] X. Li, V. Ceikute, C. S. Jensen, and K.-L. Tan. Effective online group discovery in trajectory databases. *Knowledge and Data Engineering, IEEE Transactions on*, 25(12):2752–2766, 2013.
- [13] Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–622, 2004.
- [14] J. Lin and C. Dyer. Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.
- [15] O. Uncu, W. A. Gruver, D. B. Kotak, D. Sabaz, Z. Alibhai, and C. Ng. Gridbscan: Grid density-based spatial clustering of applications with noise. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 4, pages 2976–2981. IEEE, 2006.
- [16] M. R. Vieira, P. Bakalov, and V. J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 286–295. ACM, 2009.