A Density-based Approach for Mining Movement Patterns from Semantic Trajectories

Renhe Jiang†, Jing Zhao†, Tingting Dong†, Yoshiharu Ishikawa†, Chuan Xiao‡, Yuya Sasaki§
†Graduate School of Information Science, Nagoya University, Japan
‡Institute for Advanced Research, Nagoya University, Japan
§Institute of Innovation for Future Society, Nagoya University, Japan
{jiang,dongtt,chuanx}@nagoya-u.jp, {zhao,yuya}@db.ss.is.nagoya-u.ac.jp
ishikawa@is.nagoya-u.ac.jp

***Abstract*—In this paper, we study the problem of discovering all movement patterns from semantic trajectory databases. We propose a two-step method to solve this problem efficiently. We first retrieve *frequent* movement patterns of categories from the transformed database of sequential categories, and then cluster *dense* trajectories in a growth-type way for all movement patterns. Moreover, we define a new metric distance function on trajectories. We also use M-tree to cluster trajectories more efficiently. Our experimental results demonstrate the efficiency of the proposed method.**

I. Introduction

Recently, Location-Based Services (LBSs) like Foursquare are becoming more and more popular. Due to users' check-ins at semantic spots such as shops and tourist attractions using these LBS applications, massive semantic trajectory data is being generated. While traditional trajectory data collected from GPS devices of vehicles or sensors contains temporal and spatial information, *semantic trajectory data* includes semantic information such as category additionally. For instance, a traditional trajectory of $(t_1, p_1), (t_2, p_2), (t_3, p_3)$ shows the traveling history from p_1 to p_3. A semantic trajectory with category information such as $(t_1, p_1, \mathit{Office}), (t_2, p_2, \mathit{Restaurant}), (t_3, p_3, \mathit{Office})$, contains not only the time and location information of the traveling history, but also the category information of each location.

Because of the rich information contained in semantic trajectories, the research on processing semantic trajectory data is receiving increasing attention from the database community [1], [2]. In this work, we focus on the semantic feature of category and consider the problem of mining movement patterns from semantic trajectories. This is both realistic and more meaningful, since most LBS applications associate each check-in with the category of location and we can obtain more interesting patterns based on category rather than treating each location individually.

Consider a semantic trajectory database of five objects and their trajectories as shown in Fig. 1. Each place p_i belongs to either category of Office, Shopping center and Restaurant (abbreviated as o, s and r, respectively). The minimum support is set to 3. If we treat this trajectory database as a sequence database, in other words, if we treat each p_i as an individual item, no movement pattern can be found. Instead, if we group spatially close places that belong to the same category together and treat each group as an item, a movement pattern $o \rightarrow s \rightarrow r$ can be discovered as shown in Fig. 2.

Object	Semantic Trajectory
o_1	$< p_6(s), p_5(r) >$
o_2	$< \boldsymbol{p_3(o), p_7(r), p_{10}(s)}, p_1(o) >$
o_3	$< \boldsymbol{p_2(o), p_8(r), p_{11}(s)} >$
o_4	$< \boldsymbol{p_1(o)}, p_{11}(s), \boldsymbol{p_8(r), p_{12}(s)} >$
o_5	$< p_4(o), p_9(r) >$

Fig. 1. An example semantic trajectory database

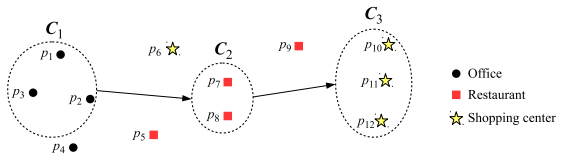


Fig. 2. An example movement pattern

In this paper, we aim at finding both *frequent* and *dense* movement patterns from semantic trajectories. In other words, the discovered patterns need to satisfy both a support threshold and a spatial closeness threshold. Our goal is to discover the union of all such patterns.

This non-trivial problem is challenging in that current approaches for mining traditional trajectory data cannot be applied directly. In [2], a method called *Splitter* is proposed to mine sequential patterns of categories in semantic trajectories. However, for each pattern, Splitter groups corresponding trajectories by decreasing a spatial parameter repeatedly, rather than based on a specified threshold. Thus, patterns discovered by Splitter tend to have different densities and are not controllable.

We propose a two-step approach for mining all movement patterns efficiently. In the first step, we detect *frequent* movement patterns of categories from the transformed database of sequential categories. At the same time, each pattern is associated with a set of semantic trajectories corresponding to this pattern. In the second step, we cluster *dense* trajectories in a growth-type way for all movement patterns. Finally, the whole set of movement patterns, each of which is associated with a subset of spatially close trajectory patterns, will be returned.

II. Preliminaries

A *semantic trajectory* is represented by a sequence of time, location and category, $(t_1, l_1, c_1), \ldots, (t_n, l_n, c_n)$. In the

978-1-4799-8641-5/15/$31.00

following,, we may call a semantic trajectory a *trajectory*.. A movement pattern of categories is called a *semantic pattern*, and represented by $C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_n$. A cluster of spatially close trajectories is called a *trajectory pattern*. A trajectory pattern corresponding to a semantic pattern is called a *semantic trajectory pattern*.

We employ and simplify the concept of snippet proposed in [2] in this work. A *snippet* $(t_1, l_1, c_1), \ldots, (t_n, l_n, c_n)$ is essentially a sub-trajectory corresponding to a semantic pattern $C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_n$, which means $c_i = C_i$. Hereafter, we use snippet as the keyword instead of trajectory. A *snippet set* is constituted of snippets corresponding to a same semantic pattern and represented by SS. Thus, all snippets within a snippet set have the same length, which is also the length of the corresponding semantic pattern. The i-th location l_i of a snippet is called the *point* of snippet at *position* i, and represented by $S.p_i$. Hereafter, a snippet is represented by a sequence of p_i, namely $p_1, p_2, \ldots, p_n$.

III. Problem Statement

In this work, we mine movement patterns based on both frequency and spatial closeness. To evaluate spatial closeness, we define a new distance function on snippets as follows:

$$d(S_1, S_2) = \max_{i=1}^{n} |S_1.p_i, S_2.p_i|, \quad (1)$$

where S_1 and S_2 are two snippets with the same length and $|\cdot, \cdot|$ is the Euclidean distance between points.

For clustering snippets, we extend DBSCAN [3], which is a popular density-based clustering algorithm that can find arbitrarily shaped clusters. It requires two parameters: the distance parameter η and the density parameter $MinObjs$. As in DBSCAN, we introduce the following concepts:

- A snippet S is a *core snippet* if at least $MinObjs$ snippets are within distance η of it, and those snippets are *directly density-reachable* from S.
- A snippet Q is called *density-reachable* from P if there is a path $S_1 S_2 \ldots S_i \ldots S_n$ with S_1 = P and S_n = Q, where each S_{i+1} is directly density-reachable from S_i.
- Two snippets P and Q are *density-connected* if there is a snippet O such that both P and Q are density-reachable from O.

The main difference from the original DBSCAN is that we use the distance function between snippets in given Eq. (1).

For example, in Fig. 3, there are four snippets, S_1, S_2, S_3 and S_4. S_1 is directly density-reachable from S_2, S_1 is density-reachable from S_3, and S_1 and S_4 are density-connected. S_1, S_2, S_3 and S_4 are mutually density-connected with each other.

Definition 1 (Density-Based Trajectory Pattern): A snippet set SS is a *density-based trajectory pattern* if the following three conditions are satisfied: (1) Size of $SS \geq MinSup$. (2) All snippets in SS are mutually density-connected w.r.t. η and $MinObjs$. (3) SS is maximal.

For example in Fig. 3, when the minimum support is 4, those four snippets can form a density-based trajectory pattern, while those four snippets in Fig. 4 cannot form a density-based trajectory pattern. Even though the minimum support is set to 3, S_2, S_3 and S_4 in Fig. 4 cannot form a density-based trajectory pattern. That is because the distance between S_2 and S_3, which is the Euclidean distance between p_{21} and p_{23}, is apparently greater than η.

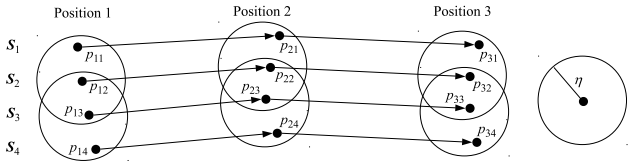


Fig. 3. Density-connected snippets ($MinObjs = 2$)

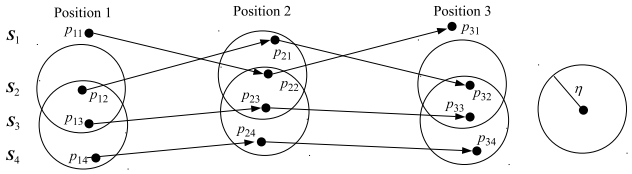


Fig. 4. Non-density-connected snippets ($MinObjs = 2$)

Discussion. So far, we have defined a density-based trajectory pattern based on the distance function on snippets. Some may doubt why the maximum Euclidean distance between corresponding points of two snippets is adopted. The reason is that based on this distance function on snippets, we can replace the second condition in Definition 1 with the following one:

> At any position i, the snapshot points of all snippets in SS are point-density-connected mutually w.r.t. η and $MinObjs$.

This property can facilitate us to find all density-based trajectory patterns efficiently. The detail is shown in Section V.

Problem Statement. Given a semantic trajectory database D, a support threshold $MinSup$, a distance threshold η and a density threshold $MinObjs$, our goal is to find all density-based semantic trajectory patterns, given by the following definition, from D.

Definition 2 (Density-Based Semantic Trajectory Pattern): Consider a length-k snippet set SS corresponding to a certain semantic pattern $C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_k$. If SS is a density-based trajectory pattern, SS is also a *density-based semantic trajectory pattern*.

IV. Density-Based Trajectory Patterns

In this section, we describe how to discover density-based trajectory patterns from a snippet set, namely, how to cluster a set of snippets using DBSCAN. Given a snippet set, a minimum support $MinSup$, a distance threshold η and a density threshold $MinObjs$, we find all density-based trajectory patterns.

A. SDBSCAN: Extension of DBSCAN for Snippets

As described in Section III, we employ an extended version of DBSCAN [3] based on the distance function on snippets. In this paper, we call the extended method *SDBSCAN*.

We would like to explain why we need SDBSCAN. The straightforward way of discovering density-based trajectory patterns using DBSCAN is as follows: (1) For each position we obtain density-connected [3] snapshot points (clusters) by

applying DBSCAN. (2) At each position, we obtain each candidate cluster to form a candidate sequence of clusters. (3) We check if each candidate sequence contains enough snippets. Fig. 4 illustrates why this method does not work for our problem. By applying DBSCAN at each position, we can get three candidate clusters, $\{p_{12}, p_{13}, p_{14}\}$, $\{p_{21}, p_{22}, p_{23}, p_{24}\}$ and $\{p_{32}, p_{33}, p_{34}\}$. The three candidate clusters form a candidate sequence of clusters and it contains three snippets S_2,S_3 and S_4. However, if $MinSup$ = 3, it still cannot form a density-based trajectory pattern as explained in Section III.

B. MSDBSCAN: Improved SDBSCAN with M-tree

It is time-consuming to perform range queries in DBSCAN, and this problem can be solved by using a spatial index R-tree. In SDBSCAN, we also need to perform range queries to find snippets within a distance of η from a given snippet. However, we cannot apply R-tree to index snippets directly because the distance function on snippets is not the Euclidean distance. We employ M-tree [4], a metric spatial indexing structure, to index snippets. The following theorem says that the distance function on snippets is a metric.

Theorem 1: The distance function Eq. (1) is a metric.

The proof in the long version of this paper [5].

In the following, we call SDBSCAN improved by M-tree *MSDBSCAN*.

V. Density-Based Semantic Trajectory Patterns

In this section, we address the problem of discovering all density-based semantic trajectory patterns from semantic trajectories. First, we introduce the naive mining method as a baseline, then we improve the method based on the Apriori property of density-based trajectory patterns.

A. Naive Mining

The naive mining method is summarized in Algorithm 1. It proceeds in two phases. In the first phase, we transform the original semantic trajectory database into a sequence database of categories. By using PrefixSpan [6], we obtain all semantic patterns and corresponding snippet sets (lines 5-6). In the second phase, for each snippet set, we apply SDBSCAN to find all candidate clusters of snippets and then check if the minimum support is satisfied (lines 10-14). Note that we can replace SDBSCAN with MSDBSCAN to improve the performance.

Algorithm 1 Naive Mining

```
1: Input: TrajectorySet, MinSup, η, MinObjs
2: Algorithm:
3: R ← ∅;                                   ▷ Result Snippet Cluster Set
4: C ← ∅;                                   ▷ Candiadate Snippet Cluster Set
5: SequenceSet ← Transform(TrajectorySet);
6: PatternSet ← PrefixSpan(SequenceSet, MinSup);
7: foreach pattern ∈ PatternSet do
8:     SnippetSet ← GetSnippetSet(pattern);
9:     C ← SDBSCAN(SnippetSet, η, MinObjs);
10:    foreach cluster ∈ C do
11:        if cluster.size ≥ MinSup then
12:            R ← R ∪ cluster;
13:        end if
14:    end for
15: end for
16: output R;
```

B. Growth-Type Mining

1) Apriori Property on Trajectory Patterns: Since our problem is to find all semantic trajectory patterns, we would like to use found short trajectory patterns for finding long ones efficiently. In fact, this idea is found in mining frequent patterns [7] known as the *Apriori property*. Next we will show that density-based trajectory patterns satisfy the Apriori property. It is based on the following lemmas.

Lemma 1: If a snippet S in a snippet set SS is a core snippet w.r.t. η and $MinObjs$, for any position i, $S.p_i$ in the point set of SS at i is a core point w.r.t. η and $MinObjs$.

Lemma 2: If two snippets X and Y in a snippet set SS are directly density-reachable w.r.t. η and $MinObjs$, for any position i, $X.p_i$ and $Y.p_i$ in the point set of SS at i are (point) directly density-reachable w.r.t. η and $MinObjs$.

Lemma 3: If two snippets X and Y in a snippet set SS are density-reachable w.r.t. η and $MinObjs$, for any position i, $X.p_i$ and $Y.p_i$ within the point set of SS at i are (point) density-reachable w.r.t. η and $MinObjs$.

Lemma 4: If snippets in a snippet set SS are density-connected, for any position i, points in the point set of SS at i are (point) density-connected w.r.t. η and $MinObjs$.

We illustrate these lemmas using Fig. 3. S_2 is a core snippet and $p_{12}(S_2.p_1)$ is a core point. S_1 is directly density-reachable from S_2, and $p_{11}(S_1.p_1)$ is (point) directly density-reachable from p_{12}. S_1 is density-reachable from S_3, and $p_{11}(S_1.p_1)$ is (point) density-reachable from $p_{13}(S_3.p_1)$. S_1, S_2, S_3 and S_4 are density-connected, while p_{11}, p_{12}, p_{13} and $p_{14}(S_4.p_1)$ are (point) density-connected. The same things hold for positions 2 and 3. Based on the lemmas, the following theorem holds.

Theorem 2: Density-based trajectory patterns satisfy the Apriori property. Namely, if snippets are density-connected w.r.t. η and $MinObjs$, corresponding sub-snippets are also density-connected w.r.t. η and $MinObjs$.

For example, in Fig. 3, the corresponding sub-snippets (from positions 1 and 3) are $S_1'\{p_{11}, p_{31}\}$, $S_2'\{p_{12}, p_{32}\}$, $S_3'\{p_{13}, p_{33}\}$ and $S_4'\{p_{14}, p_{34}\}$, respectively. S_1', S_2', S_3' and S_4' are also mutually density-connected.

2) Mining Algorithm: Next, we propose a growth-type mining method. Before the purpose, we introduce a property. We call the prefix $\{p_1, p_2, \ldots, p_{k-1}\}$ of length-k snippet $\{p_1, p_2, \ldots, p_k\}$ a *length-(k-1) snippet*. The following property holds.

Property 1: Consider a length-k snippet set SS and a length-(k-1) snippet set $SS^{'}$ constituted by the prefix of every snippet within SS. If the snippets in SS are density-connected w.r.t. η and $MinObjs$, the snippets in $SS^{'}$ are also density-connected w.r.t. η and $MinObjs$.

As an example for Property 1, in Fig. 3, four prefixes S_1' $\{p_{11}, p_{21}\}$, S_2' $\{p_{12}, p_{22}\}$, S_3' $\{p_{13}, p_{23}\}$ and S_4' $\{p_{14}, p_{24}\}$ of S_1, S_2, S_3 and S_4, are also density-connected.

The proposed algorithm in shown in Algorithm 2. There are two differences from the naive method: (1) Each pattern and its corresponding snippet set is processed in a lexical order. (2) For each pattern, we first divide snippet sets into several snippet classes according to the discovered trajectory patterns

Algorithm 2 GROWTH-TYPE MINING

```
1: Input: TrajectorySet, MinSup, η, MinObjs
2: Algorithm:
3: SequenceSet ← Transform(TrajectorySet);
4: PatternSet ← PrefixSpan(SequenceSet, MinSup);
5: SortedPatternSet ← Sort(PatternSet);
6: foreach pattern ∈ SortedPatternSet do
7:     Processing(pattern);
8: end for
9: function PROCESSING(pattern)
10:    SnippetSet ← GetSnippetSet(pattern);
11:    C ← ∅                                  ▷ Candiadate Snippet Cluster Set
12:    if pattern.prefix = null then
13:        C ← SDBSCAN(SnippetSet, η, MinObjs);
14:    else
15:        if pattern.prefix.result = ∅ then
16:            C ← ∅;
17:        else
18:            pResult ← pattern.prefix.result;
19:            SnippetClassSet ← Classify(pResult, SnippetSet);
20:            foreach class ∈ SnippetClassSet do
21:                if class.size ≥ MinSup then
22:                    C ← C ∪ SDBSCAN(class, η, MinObjs);
23:                end if
24:            end for
25:        end if
26:    end if
27:    foreach cluster ∈ C do
28:        if cluster.size ≥ MinSup then
29:            pattern.result ← pattern.result ∪ cluster;
30:        end if
31:    end for
32:    output pattern.result;
33: end function
```

(clusters of snippets), and then prune some snippets by their prefixes.

Figure 5 illustrates this idea. The snippet set corresponding to the semantic pattern $C_1 \rightarrow C_2$ is processed after the process corresponding to the semantic pattern C_1. As shown in Fig. 5, $SS = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$ is the snippet set corresponding to $C_1 \rightarrow C_2$. Since two clusters (length-1 density-based trajectory pattern) are found for C_1, we can split SS into two snippet classes: $class1 = \{S_1, S_2, S_3, S_4\}$ and $class2 = \{S_7, S_8, S_9, S_{10}\}$. Since the prefixes for S_5 and S_6 do not belong to either of the result clusters for C_1, we can directly prune them. Finally, we can apply SDBSCAN to $class1$ and $class2$ separately to improve the efficiency.

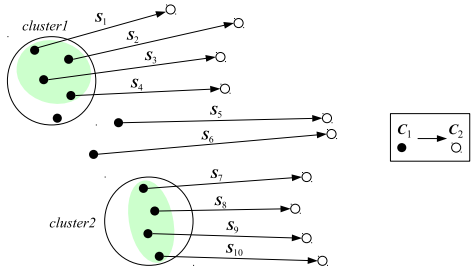


Fig. 5. An example of growth-type mining

Algorithm 2 is described as follows. As with the case of the naive method, we first transform the semantic trajectory database into a sequence database, and apply PrefixSpan [6] to obtain semantic patterns (lines 3-4). Then, we process those semantic patterns and corresponding snippet sets in a lexical order (lines 5-8). Next, if the prefix for a certain semantic pattern does not exist, which means the length of that semantic pattern is 1, we simply apply SDBSCAN to it (lines 12-13). Otherwise, we check whether the result for the prefix exists, which means some density-based trajectory patterns have been discovered for the prefix (line 15). If not, according to Property 1, we can directly prune the snippet set of that semantic pattern (line 16). Then we split the snippet set into several classes, and prune the class with a smaller size than $MinSup$. In the next step, we separately apply SDBSCAN to the left classes (lines 17-25), which is also based on Property 1. Finally, we add those candidate snippet sets that satisfy $MinSup$ into the result set (lines 27-30). Note that we can replace SDBSCAN with MSDBSCAN to improve the performance.

VI. EXPERIMENTS

A. Experimental Settings

We use both synthetic data and real data in the experiments. Synthetic trajectories are generated using the Brinkhoff's generators [8] on the road network of San Francisco. All points are in the range of [0:30,000, 0:30,000]. The length of each trajectory is either 3 or 6. We associate each of them with categories $\{A, B, C\}$ and $\{A, B, C, D, E, F\}$, respectively. The two datasets with a size of 10,000 are called S10K-3 and S10K-6, respectively.

We collect real trajectory data from check-ins of Foursquare. Three datasets, F11K-4, F11K-5 and F11K-6, are created using about $170,000$ check-ins collected from the site. The sizes of them are all 11,000 and their average lengths are 4.0, 5.0 and 6.8, respectively.

The detail of approaches used for comparison is illustrated in Table I. We evaluate all approaches based on the efficiency and the effect of the parameters.

TABLE I. LIST OF APPROACHES

Abbreviation	Approach
NaiveS	Naive approach based on SDBSCAN
NaiveM	Naive approach based on MSDBSCAN
GrowthS	Growth-type approach based on SDBSCAN
GrowthM	Growth-type approach based on MSDBSCAN

All experiments are conducted using a PC with Intel Core i7-4770 CPU (3.4GHz) and 8GB RAM running Windows 7. We implement all algorithms using Java with Eclipse.

B. Performance Analyses

1) Case Studies: First, we demonstrate some interesting trajectory patterns discovered by the proposed method using the real dataset F11K-6. As shown in Fig. 6, there are three semantic trajectory patterns corresponding to the pattern of $Restaurant \rightarrow EntertainmentSpot$. $MinSup$, η and $MinObjs$ are set to 20, 0.005 and 10, respectively. The trajectory patterns found are close to Akihabara Station, Shinjuku Station and Shibuya Station, respectively. Based on the results, we find that the movement pattern of visiting entertainment spots after dining appears frequently near the three areas illustrated using circles in Fig. 6.

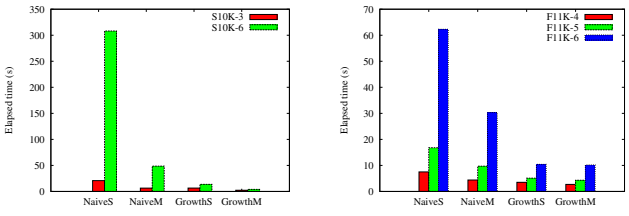
Fig. 6. Trajectory patterns of $Restaurant \rightarrow Entertainment$

Figure 7 shows two trajectory patterns. Both of them belong to the semantic pattern of *SpecializedSpot* $\rightarrow$ *Restaurant*. *SpecializedSpot* refers to specialized places such as offices, public administrations and so on. Specifically, Fig. 7(a) shows the movement pattern of moving from Pacifico Yokohama Convention Center (yellow spots) to the restaurants nearby (black spots). Besides, Fig. 7(b) illustrates the movement pattern of moving from the places close to Tokyo Big Sight (yellow spots) to the restaurants nearby Akihabara Station (black spots). Based upon the instances above, we verify that our proposed method can discover interesting semantic trajectory patterns.

(a) (b)

Fig. 7. Trajectory patterns of *SpecializedSpot* $\rightarrow$ *Restaurant*

2) Verification of Efficiency: We use S10K-3 and S10K-6 to compare the running time of the four methods. $MinSup$, η,and $MinObjs$ are set to 50, 2.0, and 1, respectively. As shown in Fig. 8(a), NaiveM is three times faster than NaiveS when the length of the trajectory is 3. The same result is obtained in the case of GrowthM and GrowthS. When the length of the trajectory is 6, the running time of NaiveS increases more rapidly than the other three methods, while the running time of GrowthS is still three times of that of GrowthM. This demonstrates the efficiency of MSDBSCAN. Since GrowthS and GrowthM are faster than NaiveS and NaiveM when the length of the trajectory is either 3 or 6, we can conclude that the growth-type method is more efficient than the naive one.



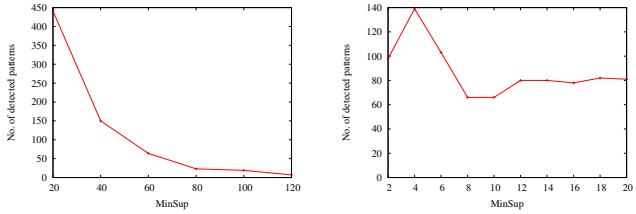
(a) (b)

Fig. 8. Running time on (a) synthetic and (b) real datasets

We also use the three real datasets F11K-4, F11K-5 and F11K-6, to compare the running time of each method. $MinSup$, η and $MinObjs$ are set to 50, 0.005, and 1, respectively. As Fig. 8(b) shows, it is almost the same result. The running time of NaiveS is two times of that of NaiveM for each dataset. Therefore, we can verify the efficiency of MSDBSCAN. Moreover, the efficiency of the growth-type method is proved by the fact that GrowthS and GrowthM are faster than NaiveS and NaiveM.

3) Effect of Parameters: We vary the parameter values to study the number of patterns found on both synthetic datasets and real datasets.

First, we evaluate all approaches by varying $MinSup$. The other two parameters are fixed as $\eta = 2.0$ and $MinObjs = 1$. We use the real dataset S10K-3. Fig. 9(a) shows the number of trajectory patterns with respect to $MinSup$. As observed in sequential pattern mining, as $MinSup$ increases, the number of trajectory patterns decreases. Then we fix the parameters as $MinSup = 50$ and $MinObjs = 1$, and observe the number of trajectory patterns with respect to η in Fig. 9(b). When varying η, the number of found trajectory patterns does not change greatly.

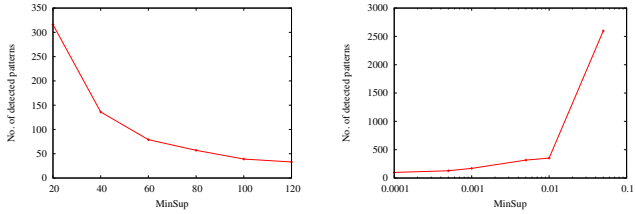


(a) Varying $MinSup$ (b) Varying η

Fig. 9. Number of detected patterns (S10K-3)

The reasons are explained as follows. The synthetic trajectory dataset S10K-3 generated by the Brinkhoff's generator is based on a road network. Thus, the generated trajectories are almost around the road segments. As a result, the density of trajectories is very high, and the number of found trajectory patterns is almost not affected.

Next, we perform a similar experiment on the real dataset F11K-6. The default settings are $\eta = 0.005$ and $MinObjs = 10$. We first vary $MinSup$. The result is shown in Fig. 10(a). As the previous experiment, the number of trajectory patterns decreases when $MinSup$ increases. Then we consider varying η by setting $MinSup = 20$ and $MinObjs = 10$. As shown in Fig. 10(b), as the distance threshold increases, the number of found trajectory patterns is also increases.



(a) Varying $MinSup$ (b) Varying η

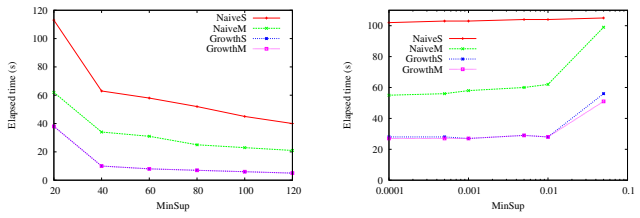
Fig. 10. Number of detected patterns (F11K-6)

We also conduct experiments on running time when varying the parameters using F11K-6. Fig. 11(a) shows the result of the running time of each approach with respect to $MinSup$. The other parameters are set as $\eta = 0.005$ and $MinObjs = 10$. Fig. 11(b) shows the result of the running time of each approach with respect to η when other parameters are set as $MinSup = 20$ and $MinObjs = 10$. Compared with Fig. 10, we can observe that the running time is affected by the number of trajectory patterns found. In other words, the higher the number of trajectory patterns found is, the slower the methods are.

C. Discussions

We summarize the experimental results below.

Firstly, we conduct the case study using the real dataset F11K-6 to find some interesting semantic trajectory patterns,



(a) Varying $MinSup$ (b) Varying η

Fig. 11. Running time on F11K-6

and verify that the methods proposed in this paper can be used to find semantic trajectory patterns effectively.

Secondly, according to the experiment on efficiency, SDB-SCAN with M-tree and the growth-type approach are verified to be more efficient. Specifically, as the length of trajectory increases, the growth-type approach is more efficient than the naive approach.

Finally, the experiments on varying parameters show that the larger the minimal support is, the less the number of trajectory patterns discovered is. In addition, the number of trajectory patterns depends on the distribution of trajectory data. Finally, we can also observe that the running time of the approaches is affected by the number of trajectory patterns found.

VII. Related Work

The emerging semantic trajectory data is attracting significant attention from the research community. [1] surveys the ideas and techniques, especially the data mining techniques, of modeling and analyzing semantic trajectories. In addition to the spatial-temporal features, recent research work on mining semantic trajectory data targets on semantic-based pattern mining. For example, Splitter [2] is proposed to mine sequential patterns in semantic trajectories. It considers both spatial-temporal and semantic features. On the other hand, [9] performs location prediction by mining trajectory patterns based on both geographic and semantic features.

On the other hand, the main focus of mining traditional trajectory data is on the spatial-temporal feature of trajectories. For example, *TRACLUS* (Trajectory Clustering) [10] is proposed for clustering trajectories. It first splits input trajectories into small segments, and then conducts clustering on those segments.

T-Pattern (Trajectory Pattern) [11] is a method for discovering trajectory patterns represented by a sequence like $RoI_0 \xrightarrow{t_1} RoI_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} RoI_n$, where RoI_i (Region of Interest) represents the discovered region passed through by a number of trajectories and t_i stands for the transition time between two RoIs. In addition, *flock* [12], *swarm* [13], *convoy* [14] and *gathering* [15] are also proposed for mining specific trajectory patterns.

MPR (Most Popular Route) [16] returns the most popular route between the given start point and end point from trajectories. In addition to the start point and the end point, *TPMFP* (Time Period-based Most Frequent Path) [17] also takes a time period as input and returns the most frequent path between the two points during that time period.

VIII. Conclusion

In this paper, we study the problem of discovering all movement patterns from a semantic trajectory database. Such a problem has not been tacked in the previous work. Towards this problem, we proposed a new metric distance function on trajectories and defined density-based semantic trajectory patterns based on the function. Moreover, we use M-tree to make trajectory clustering more efficient. In addition, we proved that density-based semantic trajectory patterns satisfy the Apriori property. Based on this property, we proposed a growth-type mining method that can discover all the semantic trajectory patterns efficiently. Finally, we used both synthetic trajectory data and real semantic trajectory data collected from Foursquare to evaluate the efficiency of our proposed method.

Acknowledgment

This research was partially supported by KAKENHI (25280039, 26540043), the MEXT program "Research and Development on Real World Big Data Integration and Analysis", and the Center of Innovation Program from JST.

References

[1] C. Parent *et al.*, "Semantic trajectories modeling and analysis," *ACM Comput. Surv.*, vol. 45, no. 4, 2013.

[2] C. Zhang *et al.*, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *PVLDB*, vol. 7, no. 9, pp. 769–780, 2014.

[3] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.

[4] P. Ciaccia *et al.*, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB*, 1997, pp. 426–435.

[5] R. Jiang *et al.*, "A density-based approach for mining movement patterns from semantic trajectories (long version)," 2015. [Online]. Available: http://www.db.ss.is.nagoya-u.ac.jp/~dbgroup/public_papers/2015-tencon-long.pdf

[6] J. Pei *et al.*, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *ICDE*, 2001, pp. 215–224.

[7] R. Agrawal and R. Srikant, "Mining sequential patterns," in *ICDE*, 1995, pp. 3–14.

[8] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[9] J. J. Ying *et al.*, "Semantic trajectory mining for location prediction," in *ACM SIGSPATIAL GIS*, 2011, pp. 34–43.

[10] J. Lee *et al.*, "Trajectory clustering: A partition-and-group framework," in *SIGMOD*, 2007, pp. 593–604.

[11] F. Giannotti *et al.*, "Trajectory pattern mining," in *KDD*, 2007, pp. 330–339.

[12] P. Laube and S. Imfeld, "Analyzing relative motion within groups of trackable moving point objects," in *GIScience*, 2002, pp. 132–144.

[13] Z. Li *et al.*, "Swarm: Mining relaxed temporal moving object clusters," *PVLDB*, vol. 3, no. 1, pp. 723–734, 2010.

[14] H. Jeung *et al.*, "Discovery of convoys in trajectory databases," *CoRR*, 2010. [Online]. Available: http://arxiv.org/abs/1002.0963

[15] K. Zheng *et al.*, "On discovery of gathering patterns from trajectories," in *ICDE*, 2013, pp. 242–253.

[16] Z. Chen *et al.*, "Discovering popular routes from trajectories," in *ICDE*, 2011, pp. 900–911.

[17] W. Luo *et al.*, "Finding time period-based most frequent path in big trajectory data," in *SIGMOD*, 2013, pp. 713–724.