Implementation and Experiments of Frequent GPS Trajectory Pattern Mining Algorithms

Xiaoliang GENG†, Hiroki ARIMURA†, and Takeaki UNO††

† Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi9, Kita–ku, Sapporo, Hokkaido, 060–0814 Japan
†† National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract Trajectory data could be analyzed through the discovery of patterns. In this paper, we propose a novel pattern named frequent envelope pattern (FEVP) of trajectory data. A frequent envelope pattern is a length--width-bounded area generated by a given trajectory data set of X. And this pattern includes all trajectory data in X. We consider the problem of searching all frequent envelope patterns. We give mining an algorithm and show the results.

Key words trajectory mining, spatio-temporal data, frequent data mining

1. Introduction

Recent years, smart mobile devices for example smart phone with GPS sensors become more and more popular. As a result, massive amount of spatio-temporal information such as real GPS stream data has attracted many researchers to extracting interesting information from it [1], [2].

In this paper, we address the problem of mining trajectory pattern from trajectory database. For solving this problem, many researchers have proposed many algorithms [1], [4], [5], [6]. These algorithms are based on flock pattern mining [1], [4] and moving cluster [5], [6]. Some of their algorithms are very effective. Their experiments support that their algorithms take good performance.

However, searching interesting trajectory pattern could be quite difficult. First, trajectory data is 2 dimensional data with time stamps. Traditional method of data mining for item set does not work. Second, trajectory data always includes time stamp such as GPS trajectory data stream. It is a problem that how to use time stamps. What is more, GPS trajectory data stream could be dense or sparse. The pruning strategy may not efficiently work, the algorithm may not judge the data correctly.

Moreover, the algorithm of mining trajectory data has to store large amount of trajectory data to analyze. As a result, this algorithm should save memory as much as possible. Trajectory data is always on-line real data stream, the algorithm should be fast enough to deal with that.

In our paper, we define the pattern as follows. Let m and n be any non-negative integers. Let $\mathcal{S} = \{ s_i \mid i = 1, \ldots, m \}$ be a collection of m two-dimensional discrete trajectories, called a *trajectory database*, whose trajectory is a sequence of 2-dim points $s_i = (p_j^i)_{j=1}^n$ having the same length n. Then, an *envelope pattern*(EVP) of width $\theta > 0$ and length $\ell \geqq 0$ is a triple $P(X) = (X, E_X, st, ed)$ of a sequence $E_X = (MBR(X_1), \ldots, MBR(X_\ell))$ of $\ell = ed - st + 1$ rectangles, and time steps $st \leqq ed$. P says that there are some set X of k moving objects that have locations close each other contained in squares $MBR(X_1), \ldots, MBR(X_\ell)$ at consecutive ℓ time stamps. Fig. 1 shows an example of an envelope pattern.

In the above definition of EVP, there can be potentially infinitely many similar patterns for the continuous nature of space domain. To overcome this problem, we then introduce the class $\mathcal{FEVP}$ of *frequent envelope patterns*, and study a mining problem for constrained frequent envelope patterns. As a main result, we present an efficient algorithm **BEM** that, given maximal width θ, minimal length ℓ, and minimal size σ, finds all frequent envelope patterns in a trajectory database. The algorithm uses depth first search to enumerate all the subsets of database and check them whether could form FEVPs. Finally, it finds out all the frequent envelope patterns.

In summary, our algorithm has the following advantages.

（1） Depth first recursive enumeration of frequent patterns,

（2） No duplication of frequent trajectory patterns,

（3） Generating any frequent pattern from database.

To examine basic characteristics of our algorithm, we do experiments and show them in Sec.4.

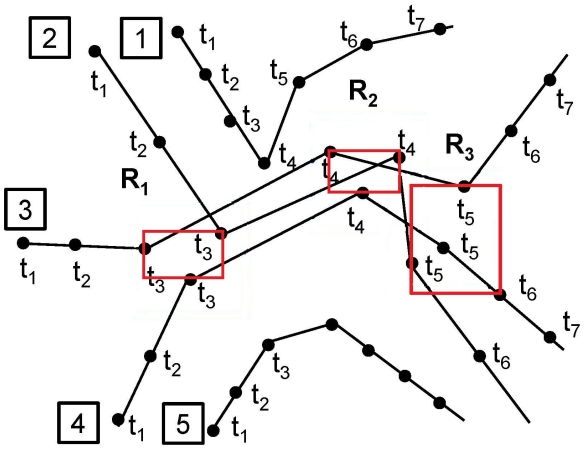

図 1 A 2-dim envelope pattern $P(X_1) = (X_1, (MBR_1, MBR_2, MBR_3), 3, 5)$ in $\mathbb{R}^2$ containing three moving objects 2, 3, 4 at time stamps t_3, t_4, and t_5 in a set of trajectories, where MBR_1, MBR_2, and MBR_3 are minimal bounding rectangles with width $\leqq \theta$. X_1 is the set of trajectory data of moving objects of 2, 3, 4.

This paper is organized as follows. Sec. 2. gives basic definitions, and Sec. 2. 4 studies frequent envelope patterns. Sec. 3. presents our algorithm and Sec. 4. shows experimental results. Finally, Sec. 5. concludes.

2. Preliminaries

In this section, we give the basic definitions and concepts on trajectory pattern mining.

2. 1 Two-dimensional discrete trajectory data

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{R}$ denote the sets of all natural and all real numbers, respectively. For any $i, j \in \mathbb{N}$ $(i \leqq j)$ and any $a, b \in \mathbb{R}$ $(a \leqq b)$, we define the intervals $[i..j] = \{i, \ldots, j\}$ and $[a, b] = \{\, x \in \mathbb{R} \mid a \leqq x \leqq b \,\}$. For a set A, A^* denotes the set of all possibly empty sequences over A. For a sequence $s = \langle a_1, \ldots, a_n \rangle$ of n elements from A, we define $|s| = n$, $s[i] = a_i$, and $s[i..j] = a_i a_{i+1} \cdots a_j$. The *empty sequence* is denoted by $\langle\rangle$.

A *point* in $\mathbb{R}^2$ is a pair $p = (x, y) \in \mathbb{R}^2$. Let s_i $(1 \leqq i \leqq m)$ be a sequence $s_i = (p_1^i, \ldots, p_n^i) = (p_j^i)_{j=1}^n$ of n points in 2-dimensional space $\mathbb{R}^2$ with sampled at time-stamps, called a *discrete trajectory* or *trajectory*, which $n \geqq 0$ denotes the length of a trajectory. $\mathcal{S} = \{s_1, \ldots, s_m\}$ is a trajectory database whose elements (points) $s_i[j] = p_j^i = (x_j^i, y_j^i) \in \mathbb{R}^2$ stands for the location of object $1 \leqq i \leqq m$ at time stamp $1 \leqq j \leqq n$ and $|\mathcal{S}| = m$. We consider i as the id of one trajectory data. Note that all trajectories in $\mathcal{S}$ have exactly same length n in our definition. For each time stamp $1 \leqq j \leqq n$, we define $S_j = \{(x_j^i, y_j^i) \mid 1 \leqq i \leqq m\}$. S_j is the set of all the points in time stamp j belonging to all trajectory data in S. Let $S_j^x = \{x_j \mid (x_j, y_j) \in S_j\}$ and $S_j^y = \{y_j \mid (x_j, y_j) \in S_j\}$ be the sets of all x- and y-coordinates of the points in S_j. We assume that $\mathcal{S}$, m, and n are fixed, otherwise stated.

2. 2 Trajectory envelope patterns

For a rectangle $R = [x_0, x_1] \times [y_0, y_1]$ in $\mathbb{R}^2$, the *width* of R is given by $width(R) = \max\{|x_1 - x_0|, |y_1 - y_0|\}$. The *minimum bounding rectangle* (MBR) containing S_j is given by $MBR(S_j) = [x_0, x_1] \times [y_0, y_1]$, where $x_0 = \min S_j^x$, $x_1 = \max S_j^x$ and $y_0 = \min S_j^y$, $y_1 = \max S_j^y$. If $S_j = \emptyset$, then $MBR(S_j) = \emptyset$, too. We know that $MBR(S_j)$ gives the unique and minimal area within all axis-parallel rectangles that contain S_j.

Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be trajectory database and $X \subseteqq \mathcal{S}$ be a subset of $\mathcal{S}$, a *2-dim trajectory envelope pattern* (or an *envelope pattern*, *EVP*) of length ℓ in $\mathcal{S}$ is a quadruple $P(X) = (X, E_X, st, ed)$, where

(1) Time stamps st and ed which $0 \leqq st \leqq ed \leqq n$ are called the *start* and *end* time such that $\ell = ed - st + 1$.

(2) $E_X = E_{st,ed}(X) = (MBR(X_{st}), \ldots, MBR(X_{st+\ell-1}))$ is a *ℓ-tuple* of 2-dim rectangles.

The *width* of $P(X)$ is $w = \max_{st \leqq j \leqq ed} width(MBR(X_j))$. We denote by $start(P(X)) = st$, $end(P(X)) = ed$, $len(P(X)) = \ell$, and $width(P(X)) = w$. If $\ell = 0$, then $P_\emptyset(X) = (X, \langle\rangle, st, st)$ is the *empty pattern*, and we define $len(P(X)) = 0$ and $width(P(X)) = 0$.

[Example 1] In Fig. 1, we show an example of an envelope pattern $P(X_1) = (X_1, (MBR_1, MBR_2, MBR_3), 3, 5)$ of length 3, such that X_1 is the set of trajectory data of moving objects of 2, 3, and 4. We see that $P(X_1)$ contains trajectory data of moving objects $2, 3$, and 4 among trajectory data of five moving objects at time stamps t_3, t_4, and t_5.

2. 3 Frequent trajectory envelope pattern

[Definition 1] We call a trajectory envelope pattern $P(X) = (X, E_X, st, ed)$ a *candidate trajectory envelope pattern* (*CandEVP*, for short) given by non-negative constraints θ and ℓ, if $P(X)$ satisfies the following conditions:

(1) $len(P(X)) \geqq \ell$,

(2) $width(P(X)) \leqq \theta$.

We call ℓ minimal length(min-length, for short) of CandEVP and call θ maximal width(max-width, for short) of CandEVP.

[Definition 2] We call a candidate trajectory envelope pattern $P(X) = (X, E_X, st, ed)$ with constraints θ and ℓ a *frequent trajectory envelope pattern* (*FEVP*, for short) given by a non-negative constraint σ, if $P(X)$ satisfies $\mid X \mid \geqq \sigma$. We call σ minimal size(min-size, for short) of CandEVP.

[Definition 3] For any $1 \leqq st \leqq ed$, we denote by $\mathcal{FEVP}(\mathcal{S}, st, \sigma, \theta, \ell)$, the class of all frequent envelope patterns *FEVP* with starting position st, $X \subseteqq S$, and non-negative constraints σ, θ, and ℓ.

Algorithm 1 The main algorithm for finding all frequent envelope patterns

```
1: procedure main(S, θ, ℓ, σ)
2:     Θ ← (θ, ℓ, σ);
3:     for i ← 1, ..., n − ℓ + 1 do
4:         start ← i
5:         for c ← 1, ..., m do
6:             BEM(c, ∅, start, S, Θ);
7:         end for
8:     end for
9: end procedure
```

Algorithm 2 Basic frequent envelope trajectory patterns

```
1: procedure BEM(c, X, start, S, Θ)
2:     Y ← X;
3:     for i ← S[s_c, ..., s_m] do
4:         Y ← Y ⋃{i}
5:         if CheckCandEVP(Y, start, S, Θ) then
6:             get θ from Θ
7:             if | Y |≧ θ then
8:                 Output Y as FEVP;
9:             end if
10:            BEM(i, Y, start, S, Θ)
11:        end if
12:    end for
13: end procedure
```

2.4 Frequent pattern mining problem

The number of all possible FEVPs in a given database $\mathcal{S}$ may be prohibitive. Thus, we incorporate a set of constraints as well as closeness into our envelope pattern mining problem. Now, we state our problem as follows.

Frequent Envelope Pattern Mining Problem:
Input: A collection $\mathcal{S} = \{s_1, \ldots, s_m\}$ of 2-dim trajectories with real numbers of max-width $\theta > 0$, min-length $\ell > 0$, and min-size $\sigma \geqq 1$.
Output: Find all frequent envelope patterns $P(X)$ in $\mathcal{FEVP}(\mathcal{S}, st, \theta, \ell, \sigma)$ $(X \subseteqq \mathcal{S})$ appearing in $\mathcal{S}$ such that $width(P(X)) \leqq \theta$, $len(P(X)) \geqq \ell$, and $\mid X \mid \geqq \sigma$ without duplicates.

3. Proposed Algorithm

We present an algorithm **BEM** (Basic envelope pattern miner) for mining frequent envelope trajectory patterns under given constraints from an input trajectory database.

3.1 Outline of our algorithm

In Alg.1, Alg.2 and Alg.3, we present the main algorithm **main**, its recursive sub-procedure **BEM** and checking CandEVP algorithm **CheckCandEVP** respectively.

The main algorithm **BEM** in Alg.1 first receives an input trajectory database $\mathcal{S}$ and constraint parameters $\theta > 0$, $\ell > 0$, and $\sigma \geqq 1$. Next it invokes the recursive sub-procedure **BEM** in Alg.2 with each singleton trajectory data c and other arguments in line 6. BEM algorithm enumerates all subsets of $\mathcal{S}$ and calls the algorithm **CheckCandEVP** to check whether these subsets could form CandEVPs.

Algorithm 3 Check CandEVP algorithm

```
1: procedure CheckCandEVP(X, start, S, Θ)
2:     Y ← X;
3:     get σ from Θ
4:     get ℓ from Θ
5:     if len(Y) < ℓ then
6:         return FALSE
7:     end if
8:     for i ← Y[1, ..., len(Y)] do
9:         if width(MBR(Y_i)) > σ then
10:            return FALSE
11:        end if
12:    end for
13:    return TRUE
14: end procedure
```

3.2 Basic envelope trajectory pattern miner algorithm BEM

3.2.1 Enumerate subsets of $\mathcal{S}$

Algorithm **BEM** depth first enumerates subsets of $\mathcal{S}$. For every subset, this algorithm calls CheckCandEVP in line 5 to check whether it could form a CandEVP. If this subset is a CandEVP, **BEM** checks the size of it. If the size is no less than θ, this subset is outputted as a FEVP; else, recursively call **BEM** itself for further searching. If this subset is not a CandEVP, this algorithm prunes this subset and requires a new subset.

3.2.2 Call CheckCandEVP

Algorithm **CheckCandEVP** checks a subset X of trajectory data with the constrains of $start, \mathcal{S}, \theta, \ell$. It computes the *MBR* from *start* time stamp to the end of trajectory data or the computation stops at the time stamp where the $width(MBR) > \theta$. Next, this algorithm computes the length by $end - start + 1$. If the length $\geqq \ell$, it returns TRUE; else it returns FALSE.

3.3 Analysis

Combining the above arguments, we give the correctness and the complexity of our main algorithm. Let m be the *size* of input database S and n be the *length* of trajectory data. The **main** algorithm search from start time stamp of trajectory data to $n - \ell + 1$. Therefore, the complexity of **main** is $O(n)$. The enumeration of subsets of trajectory data taken by **BEM** and **main** enumerates 2^m subsets of S at most. For checking them one by one, the complexity of **CheckCandEVP** is $O(n \log m)$. Clearly, the total complexity is $O(m2^m n \log m)$.

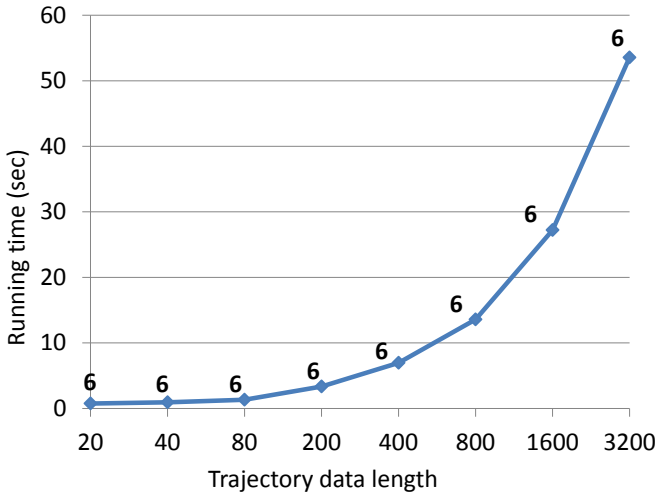

図 2 The result on the running time (left) by varying the length of trajectory data in database. Min-size is 3, min-length is 12, max-width is 1. DB size is 100, pattern quantity is 6, pattern size is 3, pattern width is 1, pattern length is 12, pattern start is 3.

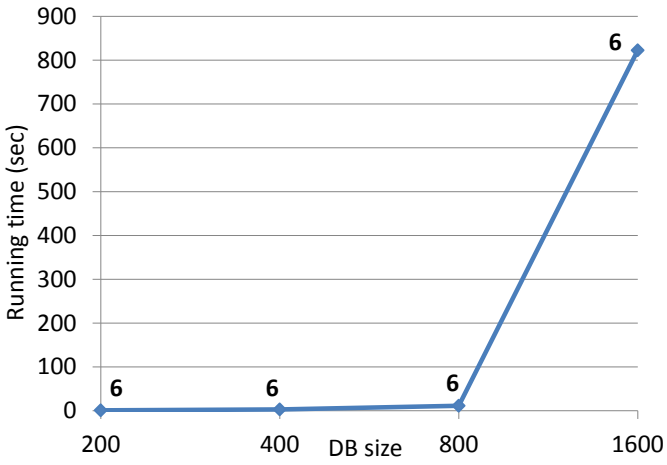

図 3 The result on the running time (left) by varying the DB size of trajectory data in database. Min-size is 3, min-length is 12, max-width is 1. Trajectory data length is 20, pattern quantity is 6, pattern size is 3, pattern width is 1, pattern length is 12, pattern start is 3.

4. Experiments

Finally, we report preliminary computational experiments to examine basic properties of the proposed algorithm **BEM** on trajectory data-set.

4.1 Data and method

We used random trajectory data-set generated by *Data Generator*. Data Generator is a program we made for validating and testing our algorithm. It generates trajectory data as "x coordinate, y coordinate, time stamp" given by parameters of quantity, length, time interval of trajectory data and the quantity, size, width, start, length of patterns in trajectory data and the min-bound and max-bound of random x and y coordinate of trajectory data. We set time interval is 1 second, min-bound is 1, and max-bound is 10 for all experiments. The following is an example of four lines of one trajectory data file:

8.85633,6.80149,1
4.91046,3.70516,2
5.2009,9.26173,3
4.50338,3.21286,4
...

All the lines in one file belong to one moving object.

We implemented our algorithm **BEM** in C++ and compiled by g++ of GNU, version 4.5.3. We used a PC (Intel Core i7 CPU, 2.80GHz, 8GB of RAM) running Windows 7.

4.2 Results

In Fig. 2, we show the result on the running time (left) by varying the length of trajectory data in database. We observed that the running time increases as the trajectory

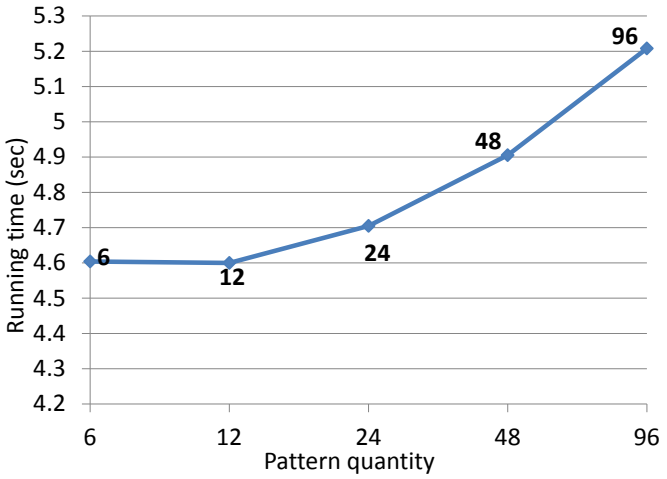

図 4 The result on the running time (left) by varying the pattern quantity of trajectory data in database. Min-size is 3, min-length is 12, max-width is 1. Trajectory data length is 20, DB size is 500, pattern size is 3, pattern width is 1, pattern length is 12, pattern start is 2.

data length increases, and for every experiment, the algorithm could find all 6 patterns included in database.

In Fig. 3 and Fig. 4, we show the result on the running time (left) by varying the size of database and quantity of patterns included in database. We observe that the running time increases as these two parameters increases, and for every experiment, the algorithm could find all patterns included in database.

In Fig. 6, Fig. **??** and Fig. 7, we show the results on the running time (left) and the quantity of patterns found by our algorithm (near the points) by varying the min-size, the max-width, and the min-length, respectively. In these figures, we

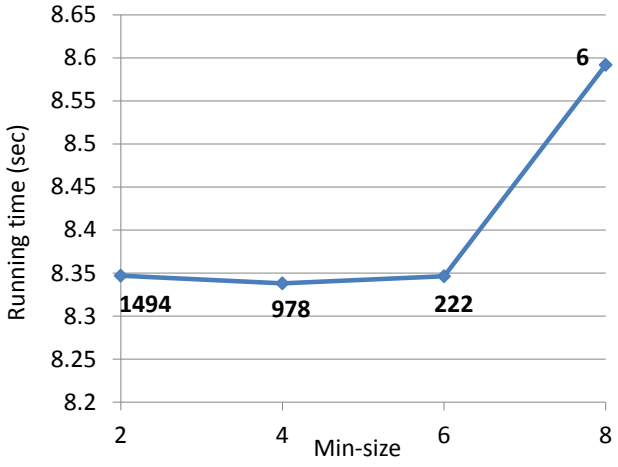

図 5 The result on the running time (left) by varying the input of min-size. Min-length is 12, max-width is 1. Trajectory data length is 20, DB size is 500, pattern size is 8, pattern quantity is 6, pattern width is 1, pattern length is 12, pattern start is 2.

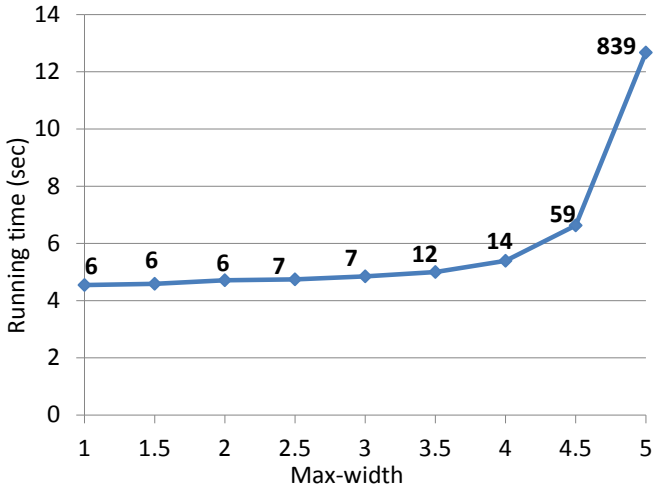

図 6 The result on the running time (left) by varying the input of max-width. Min-length is 12, min-size is 3. Trajectory data length is 20, DB size is 500, pattern size is 3, pattern quantity is 6, pattern width is 1, pattern length is 12, pattern start is 2.

first saw that the algorithm discovered the sub-patterns included in bigger longer ones.

Summary of Experiments: As it can be seen from the figures, our algorithm can find all patterns included in database successfully in Fig. 2, Fig. 3 and Fig. 4. But it also finds sub-patterns in Fig. 6 and Fig. 7 when we input smaller min-size and shorter min-length. As the input parameters of min-size and max-width become larger in Fig. 6 and Fig. **??**, the performance of our algorithm turns bad obviously.

5. Conclusion

We presented an efficient algorithm **BEM** that, given max-

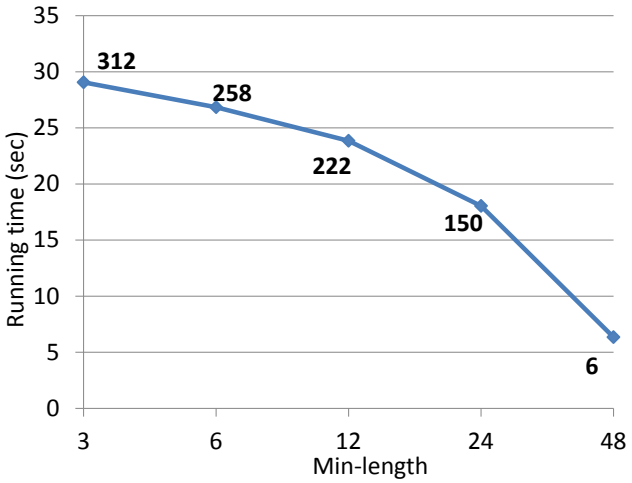

図 7 The result on the running time (left) by varying the input of min-length. Max-width is 1, min-size is 3. Trajectory data length is 60, DB size is 500, pattern size is 3, pattern quantity is 6, pattern width is 1, pattern length is 48, pattern start is 2.

width θ, min-length ℓ, and min-frequency σ, finds all frequent envelope patterns with width $\leqq \theta$, length $\geqq \ell$, and frequency $\geqq \sigma$ in a trajectory database.

文　献

[1] V. Marcos, B. Petko, and V. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proc. ACM GIS'09*. ACM, 2009, pp. 286–295.

[2] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proc. KDD'07*. ACM, 2007, pp. 330–339.

[3] J. Hoyoung, L. Y. Man, Z. Xiaofang, J. Christian, and S. HengTao, "Discovery of convoys in trajectory databases," in *Proc. PVLDB'08*. VLDB Endowment, 2008.

[4] M. Benkert, J. Gudmundsson, F. Hubner, and T. Wolle, "Reporting flock patterns," *Computational Geometry*, vol. 41, pp. 111–125, 2008.

[5] L. Zhenhui, D. Bolin, H. Jiawei, and K. Roland, "Swarm: Mining relaxed temporal moving object clusters," in *Proc. PVLDB'10*. VLDB Endowment, 2010.

[6] K. Panos, M. Nikos, and B. Spiridon, "On discovering moving clusters in spatio-temporal data," in *Proc. SSTD'05*, ser. LNCS, vol. 3633. Springer-Verlag, 2005.